NASA CR-178457

A Reproduced Copy

OF

N85 - 32793

RESEARCH ON AN EXPERT SYSTEM FOR DATABASE OPERATION OF SIMULATION/EMULATION MATH MODELS

Phase I Results

VOLUME II

Prepared by:

K. Kawamura, G.O. Beale, J.D. Schaffer,
B.-J. Hsieh, S. Padalkar, J.J. Rodriguez-Moscoso

Center for Intelligent Systems
Vanderbilt University
P.O. Box 1804, Station B
Nashville, Tennessee 37235

This work was performed for NASA's George C. Marshall Space Flight Center under contract NAS8-36285.

AUGUST 9, 1985

RESEARCH ON AN EXPERT SYSTEM FOR DATABASE OPERATION OF SIMULATION/EMULATION MATH MODELS

Phase I Results

VOLUME II

Prepared by:

K. Kawamura, G.O. Beale, J.D. Schaffer,
B.-J. Hsieh, S. Padalkar, J.J. Rodriguez-Moscoso

Center for Intelligent Systems
Vanderbilt University
P.O. Box 1804, Station B
Nashville, Tennessee 37235

This work was performed for NASA's George C. Marshall Space Flight Center under contract NAS8-36285.

AUGUST 9, 1985

VOLUME II

TABLE OF CONTENTS

APPENDICES

Appendix A A.1
NESS User's Manual

Appendix B B.1
NESS Programmer's Manual

Appendix C C.1
Theoretical Background on Quaternion

Appendix D D.1
NESS Listing
Simulation Model Listing

APPENDIX A

APPENDIX A

NESS USER'S MANUAL

This is a reference manual for NESS, a simulation expert-system developed at Vanderbilt University. This manual will give user information regarding starting and operating NESS.

1. STARTUP

To start NESS one must log onto the VAX 11/780 at Vanderbilt. The VMS prompt '$' appears on the screen. The user then executes the following commands:

```
$ ness
$ shell                 [This places you under "EUNICE" a UNIX-like operating
                         system.]
% lisp                  [% is EUNICE's prompt. LISP gets you into the FRANZLISP
                         environment.]
Franz Lisp, Opus 38.79

Do you want to run:

   1)  NASA EXPERT Simulation SYSTEM ?
   2)  GENIE (GENeric Inference Engine) ?
   3)  Franz Lisp ?

Please enter choice(s):
   # 1
Before running the NASA Expert Simulation System,
did you type in 'SHELL' while you were under VMS?

   1)  yes
   2)  no
   3)  don't know

Please enter choice(s):
   # 1
============================================================
:      Welcome to NASA Expert Simulation System (NESS)     :
============================================================

Loading GENIE, and NESS....
GENIE version 2.3 generated on Tues. Apr. 2 12:25:06 1985
NESS version 1.0 generated on Thu. Jun. 27 11:45:05 1985
   [be patient this may take one or two minutes]
```

This expert system provides an intelligent interface to a generic simulation program for spacecraft attitude control problems. Below is a menu of the functions the system can perform. Control will repeatedly return to this menu after executing each user request. When you are ready for further text display, hit the 'return' key:

2. NESS OPERATION

Now you are in the control of NESS. Its TOP LEVEL MENU will appear on the screen. Please make only one choice at a time. The menu appears as follows:

```
top_level_choice
  top_level_menu

              1)  Exit to GENIE
              2)  Set up initial parameter values
              3)  Run simulation program
              4)  Display current parameter values required for simulation
              5)  Display outputs generated by simulation
              6)  Change initial parameter values required for simulation
              7)  Set up initial parameter values to default values
              8)  Store current parameter values in a disk-file
```

Please enter choice(s). Now you have to enter the number of the function you want to be performed. You are strongly advised to enter only one choice at a time.

A brief explanation of what happens when each of the choices is selected follows.

1) Exit to GENIE

You can use this function to halt NESS and go to the GENIE environment. To return to NESS use option 2 of GENIE's menu. This is not advised for those not familiar with GENIE.

2) Set up initial parameter values

This function allows you to enter initial values of parameters necessary for the simulation experiment. NESS will ask you some questions to infer or to provide values.

An option is provided here to allow users to run both types of responses with the same values for parameters. The first question asked by NESS concerns this aspect.

NESS assumes the inertial matrix to be diagonal with all diagonal elements equal. Hence it prompts the user for only one diagonal element of the inertial matrix.

NESS asks some questions to get values of the Kp and Kd controller matrices. If proportional control is not desired then all elements of the Kp matrix are set to 0.0. Likewise if differential control is not wanted then all elements of the Kd matrix are set to 0.0. If no cross-coupling is desired the Kp and Kd become diagonal matrices, otherwise, the user is asked whether he wants them to be symmetric or regular. If the user wants the same controller gain for all axes in a diagonal matrix, then NESS prompts the user for only one element of that particular matrix. Depending

on the reply from the user, NESS configures the Kp and Kd matrices.

If the user wants the Quaternion block to be included in the simulation, then NESS prompts for initial values of the Quaternion.

NESS asks the user to choose a system response type from a menu of responses (step response or frequency response).

Some questions are asked about initializing angular velocity and angular position. They include whether the user wants all 0.0 initial values, the same initial values for all axes, or different initial values for the three axes.

For other parameters NESS simply asks the user to enter values. NESS knows about the parameters required for each response type, and only asks for the values of the required parameters.

3) Run simulation program

After using option 2 to set initial values of parameters, the user can use this option to run the the simulation program. This function first calculates eigenvalues of the system matrix. From the eigenvalues it infers the value of Tfinal for step response (from some heuristics). It displays this value to the user, and, if the user wants to have a different value for Tfinal, he can override this value. For the other two types of responses, NESS displays the value of TAU or the time constant of the system. The FORTRAN program gets this value, and, by using some heuristics, it calculates Tfinal, for each frequency separately.

If either Kp or Kd or both contain all 0.0 elements, NESS warns the user about this. In this case, NESS goes to a default value of TAU (default = 1.0).

Finally the user has the option of not running the simulation program. If he wants to return to the TOP LEVEL MENU to review and alter some parameter values, he can do so.

If the user wants to run the simulation program, the FORTRAN program is executed. After execution, control is returned to the TOP LEVEL MENU.

If the user picks this option, (i.e., run simulation program before values have found for parameters) then NESS simply displays a message and returns to the user to set up initial parameter values.

4) Display current parameter values required for simulation

Using this option the user can look at parameter values present in the database of NESS. A menu of parameters known to NESS is displayed and the user can choose the parameters whose values are to be displayed. If the parameter has a value it is displayed, else "nil (empty frame level)" is displayed signifying that that particular parameter does not have any value. The user can return to the TOP LEVEL MENU using option 1 (Return to TOP level) present on this menu.

If none of the parameters have any values then NESS displays a message to that effect.

5) Display outputs generated by simulation

This option allows the user to look at the results of the simulation experiment. A menu of outputs generated by the simulation is displayed and the user can choose the output that is to be displayed. The outputs generated by the simulation program are: plot of omega, plot of theta, numerical outputs and characteristics of the analysis. The user can return to the TOP LEVEL MENU by choosing option 1 on this menu.

NESS will not display any outputs if the simulation program has not been run, or has not executed properly. It will display a message to that effect to the user when he selects this option.

6) Change initial parameter values required for simulation

This option allows the user to change parameter values. A menu of parameters known to NESS is displayed and the user can choose the parameters whose values are to be changed. NESS then asks the user for the new value of that parameter. The user can return to the TOP LEVEL MENU by choosing option 1 present on this menu.

Let it be noted that the value entered by the user is stored in the database by NESS without checking its validity. Thus a non-numerical value can be entered for a parameter, leading to an error later on. This error will be detected by the FRANZLISP interpreter when it tries to create the data file of parameter values for the FORTRAN program. The user will have to do his own checking to find out the wrongly entered value. Hence this option must be used with caution. If an error occurs, it is best to use option 2 of the TOP LEVEL MENU to gather values for parameters.

The user will not be allowed to change values of parameters, if none of them have any value. This message will be displayed to the user when he selects this option.

7) Set up initial parameter values to default values

The user can use values stored in a disk file to be be entered in NESS's database by selecting this option. NESS displays the current contents of the current subdirectory and asks the user to enter the file containing initial values. The file entered by the user must have been created by NESS using option 8 of the TOP LEVEL MENU. Any other file must not be entered. If this is done unexpected errors might occur.

8) Store current parameter values in a disk file

The user can store current parameter values in NESS's database to a disk file.The stored values can be used for later experimentation by using option 7 of the top level menu. NESS displays the current contents of the current subdirectory and asks the user to enter a filename, where the current initial values will be stored. The user must not enter a filename already existing in the current subdirectory unless he is sure that doing

so will not endanger a file crucial to NESS. The files crucial to NESS are given below.

simxpert.l, rules.l, change.l, user.l, demonar.l,
lisprc.l, demon00.l, demon01.l, demon02.l,
demon03.l, eigenval.o and main.exe.

ENGLISH VERSION OF THE RULES IN VARIOUS RULE-BASES

1) **value_input_rb:**

```
Rule_1:   IF    [all simulation parameters are known]

          THEN  [run the simulation]

Rule_2:   IF    [the inertial matrix is full]
            and [the Kp matrix is full]
            and [the Kd matrix is full]
            and [the Quaternion is initialized]
            and [an integration method is found]
            and [the type of response is known]
            and [the axis of input command is known]
            and [the value of T0 is known]
            and [the value of error is known]
            and [the Y matrix is full]

          THEN  [the first_step parameters are known]

Rule_3:   IF    [step_response was chosen]
            and [first_step_parameters are known]
            and [the value of deltaT is known]
            and [the value of steady state error is known]
          or    [frequency_response was chosen]
            and [first_step_parameters are known]
            and [second_step_parameters are known]

          THEN  [all parameters are known]

Rule_4:   IF    [the value of amplitude is known]
            and [the value of initial_freguency is known]
            and [the number_of_samples_per_period is known]
            and [the value of number_of_decades is known]
            and [the value of number_of_sampling_frequency_per_decade
                 is known]
            and [the value of phase is known]

          THEN  [the second_step parameters are known]

Rule_5:   IF    [derivative control is not desired]

          THEN  [Kd_matrix_type is zero]
            and [Kd_matrix is full]
            and [each element of Kd_matrix=0]

Rule_6:   IF    [proportional control is not desired]

          THEN  [Kp_matrix_type is zero]
            and [Kp_matrix is full]
            and [each element of Kp_matrix=0]

Rule_7:   IF    [cross-coupling_between_axes for proportional control is
                 not desired]
```

```
          THEN  [Kp_matrix_type is diagonal]
            and [each off-diagonal element of Kp_matrix=0]

Rule_8:   IF    [cross-coupling_between_axes for differential control is
                 not desired]

          THEN  [Kd_matrix_type is diagonal]
            and [each off-diagonal element of Kp_matrix=0]

Rule_9:   IF    [cross-coupling_between_axes for proportional control is
                 not desired]
            and [the same proportional control is desired for all axes]
            and [the value of Kp(1,1) is known (user will be asked)]

          THEN  [Kp_matrix_type is equal_diagonal_terms]
            and [each diagonal element of Kp_matrix=Kp(1,1)]

Rule_10:  IF    [cross-coupling_between_axes for differential control is
                 not desired]
            and [the same differential control is desired for all axes]
            and [the value of Kd(1,1) is known (user will be asked)

          THEN  [Kd_matrix type is equal_diagonal_terms]
            and [each diagonal element of Kd_matrix=Kd(1,1)]

Rule_11:  IF    [cross-coupling_between_axes for proportional control is
                 not desired]
            and [the same proportional control is not desired for all axes]
            and [the value of Kp(1,1) is known (user will be asked)]
            and [the value of Kp(2,2) is known (user will be asked)]
            and [the value of Kp(3,3) is known (user will be asked)]

          THEN  [Kp_matrix_type is not equal_diagonal_terms]

Rule_12:  IF    [cross-coupling_between_axes for differential control is
                 not desired]
            and [the same differential control is not desired for all axes]
            and [the value of Kd(1,1) is known (user will be asked)]
            and [the value of Kd(2,2) is known (user will be asked)]
            and [the value of Kd(3,3) is known (user will be asked)]

          THEN  [Kd_matrix_type is not equal_diagonal_terms]

Rule_13:  IF    [the value of inertial_matrix (1 1) is known (user will be
                 asked)]

          THEN  [inertial_matrix (2 2) = inertial_matrix (1 1)]
            and [inertial_matrix (3 3) = inertial_matrix (1 1)]
            and [inertial_matrix is full]
            and [each off-diagonal element of inertial_matrix = 0]

Rule_14:  IF    [cross-coupling_between_axes for proportional control
                 is desired]
            and [the Kp matrix is symmetric]
            and [the value of Kp_matrix(1,1) is known (user will be asked)]
            and [the value of Kp_matrix(1,2) is known (user will be asked)]
```

```
          and [the value of Kp_matrix(1,3) is known (user will be asked)]
          and [the value of Kp_matrix(2,2) is known (user will be asked)]
          and [the value of Kp_matrix(2,3) is known (user will be asked)]
          and [the value of Kp_matrix(3,3) is known (user will be asked)]

        THEN  [Kp_matrix_type is symmetric]
          and [Kp(2,1)=Kp(1,2)]
          and [Kp(3,1)=Kp(1,3)]
          and [Kp(3,2)=Kp(2,3)]

Rule_15:  IF    [cross-coupling_between_axes for differential control is
                 desired]
          and [the Kd_matrix is symmetric]
          and [the value of Kd_matrix(1,1) is known (user will be asked)]
          and [the value of Kd_matrix(1,2) is known (user will be asked)]
          and [the value of Kd_matrix(1,3) is known (user will be asked)]
          and [the value of Kd_matrix(2,2) is known (user will be asked)]
          and [the value of Kd_matrix(2,3) is known (user will be asked)]
          and [the value of Kd_matrix(3,3) is known (user will be asked)]

        THEN  [Kd_matrix_type is symmetric]
          and [Kd(2,1)=Kd(1,2)]
          and [Kd(3,1)=Kd(1,3)]
          and [Kd(3,2)=Kd(2,3)]

Rule_16:  IF    [cross-coupling_between_axes for proportional control is
                 desired]
          and [the Kp_matrix is not symmetric]
          and [the value of Kp_matrix(1,1) is known (user will be asked)]
          and [the value of Kp_matrix(1,2) is known (user will be asked)]
          and [the value of Kp_matrix(1,3) is known (user will be asked)]
          and [the value of Kp_matrix(2,1) is known (user will be asked)]
          and [the value of Kp_matrix(2,2) is known (user will be asked)]
          and [the value of Kp_matrix(2,3) is known (user will be asked)]
          and [the value of Kp_matrix(3,1) is known (user will be asked)]
          and [the value of Kp_matrix(3,2) is known (user will be asked)]
          and [the value of Kp_matrix(3,3) is known (user will be asked)]

        THEN  [Kp_matrix_type is regular]

Rule_17:  IF    [cross-coupling_between_axes for differential control
                 is desired]
          and [the Kd_matrix is not symmetric]
          and [the value of Kd_matrix(1,1) is known (user will be asked)]
          and [the value of Kd_matrix(1,2) is known (user will be asked)]
          and [the value of Kd_matrix(1,3) is known (user will be asked)]
          and [the value of Kd_matrix(2,1) is known (user will be asked)]
          and [the value of Kd_matrix(2,2) is known (user will be asked)]
          and [the value of Kd_matrix(2,3) is known (user will be asked)]
          and [the value of Kd_matrix(2,3) is known (user will be asked)]
          and [the value of Kd_matrix(2,3) is known (user will be asked)]
          and [the value of Kd_matrix(3,1) is known (user will be asked)]
          and [the value of Kd_matrix(3,2) is known (user will be asked)]
          and [the value of Kd_matrix(3,3) is known (user will be asked)]

        THEN  [Kd_matrix_type is regular]
```

```
Rule_18:  IF    [the Kp_matrix_type is diagonal]
            and [the Kp_matrix has equal diagonal terms]
          or    [the Kp_matrix_type is diagonal]
            and [the Kp_matrix has unequal diagonal terms]
          or    [the Kp_matrix_type is symmetric]
          or    [the Kp_matrix_type is regular]

          THEN  [Kp_matrix is full]

Rule_19:  IF    [the Kd_matrix_type is diagonal]
            and [the Kd_matrix has equal diagonal terms]
          or    [the Kd_matrix_type is diagonal]
            and [the Kd_matrix has unequal diagonal terms]
          or    [the Kd_matrix_type is symmetric]
          or    [the Kd_matrix_type is regular]

          THEN  [Kd_matrix is full]

Rule_20:  IF    [the Quaternion is wanted by the user]
            and [the value of roll_angle is known]
            and [the value of pitch_angle is known]
            and [the value of yaw_angle is known]

          THEN  [the Quaternion is initialized]
            and [initial value of theta for all axes to be 0.0 is wanted]

          ELSE  [the Quaternion is initialized]

Rule_21:  IF    [initial value of theta for all axes to be 0.0 is wanted]

          THEN  [the value of tcontrol_1 is true]
            and [theta_x = theta_y = theta_z = 0.0]

Rule_22:  IF    [initial value of theta for all axes to be equal is wanted]
            and [the value of theta_x is known]

          THEN  [the value of tcontrol_2 is true]
            and [theta_y = theta_x]
            and [theta_z = theta_x]

Rule_23:  IF    [initial values of theta for all axes to be unequal is wanted]
            and [the value of theta_x is known]
            and [the value of theta_y is known]
            and [the value of theta_z is known]

          THEN  [the value of tcontrol_3 is true]

Rule_24:  IF    [the value of tcontrol_1 is found to be true]
            or  [the value of tcontrol_2 is found to be true]
            or  [the value of tcontrol_3 is found to be true]
          and   [the value of ocontrol_1 is found to be true]
            or  [the value of ocontrol_2 is found to be true]
            or  [the value of ocontrol_3 is found to be true]

          THEN  [the Y matrix is full]
```

```
Rule_25:  IF    [initial value of omega for all axes to be 0.0 is wanted]

          THEN  [the value of ocontrol_1 is true]
            and [omega_x = omega_y =omega_z = 0.0]

Rule_26:  IF    [initial values of omega for all axes to be equal is wanted]
            and [the value of omega_x is known]

          THEN  [the value of ocontrol_2 is true]
            and [omega_y = omega_x]
            and [omega_z = omega_x]

Rule_27:  IF    [initial values of omega for all axes to be unequal is wanted]
            and [the value of omega_x is known]
            and [the value of omega_y is known]
            and [the value of omega_z is known]

          THEN  [the value of ocontrol_3 is true]

Rule_29:  IF    [integration method selected is Euler]

          THEN  [the value of error is 0.0]
```

2) **run_rb:**

```
Rule_1:   IF    [response type chosen is step response]
            and [the value of Tfinal has been found]
          or    [response type chosen is frequency response]
          and   [the user wants to run the simulation program]

          THEN  [run the simulation program]

Rule_2:   IF    [response type chosen is step response]
            and [user likes the calculated value of Tfinal]

          THEN  [the value of Tfinal has been found]

          ELSE  [ask user to provide his value for Tfinal]
            and [the value of Tfinal has been found]
```

3) **output_display_rb:**

All rules in this rule_base are of the same form viz

```
          IF    [user wants to observe a particular output from
                 the simulation program]

          THEN  [display that output on the screen]
```

4) **disp_init_val_rb:**

All rules in this rule_base are of the same form viz

```
          IF    [user wants to observe the value of a particular parameter]

          THEN  [display the value of that particular parameter on the screen]
```

5) **change_param_rb:**

All rules in this rule_base are of the same form viz

IF [user wants to change the value of a particular parameter]

THEN [ask the user to provide his value for that particular parameter]
and [set that particular parameters value to the value given by the user]

6) **change_matrix_rb:**

Rule_1: IF [user wants to change the value of an element of a matrix]

THEN [ask user for row column and value]
and [set the element in the given row and column to the given value]

7) **stage_1_rb:**

Rule_1: IF [user wants to observe outputs generated by the simulation program]
and [simulation run has been successful]

THEN [display a menu of the various outputs generated by the simulation]

Rule_2: IF [user wants to run the simulation program]
and [all simulation parameter values are known]

THEN [run the simulation program]

Rule_3: IF [user wants to observe values of parameters]
and [all simulation parameter values are known]

THEN [display a menu of the various parameters known to NESS]

Rule_4: IF [user wants to store parameter values in the database of NESS to a file]
and [all simulation parameter values are known]

THEN [store parameter values to file whose name is supplied by user]

Rule_5: IF [user wants to run the simulation program]
and [all simulation parameter values are not known]

THEN [display message saying user cannot run simulation program]

Rule_6: IF [user wants to set parameter values to default values]

THEN [set parameter values to values found in file whose name is supplied by user]

Rule_7: IF [user wants to change values of parameters]

and [all simulation parameter values are known]

THEN [display a menu of the parameters known to NESS]

Rule_8: IF [user wants to set up parameter initial values]
and [all simulation parameter values are known]

THEN [run the user_rb]

Rule_9: IF [user wants to observe the outputs generated by the simulation program]
and [the simulation run has not been successful]

THEN [display a message telling the user that outputs cannot be displayed]

Rule_10: IF [user wants to change values of parameters]
and [all simulation parameter values are not known]

THEN [display a message telling the user that parameter values cannot be changed]

Rule_11: IF [user wants to observe values of parameters]
and [all simulation parameter values are not known]

THEN [display a message telling the user that he cannot observe the values of parameters]

Rule_12: IF [user wants to set up parameter initial values]
and [all simulation parameter values are not known]

THEN [run the value_input_rb]

Rule_13: IF [user wants to store parameter values in the database of NESS to a file]
and [all simulation parameter values are not known]

THEN [display a message telling the user that parameter values in the database of NESS cannot be stored in a file]

8) **user_rb:**

Rule_1: IF [user wants to change current response type]

THEN [all current parameter values will be saved and user will be prompted for a new response type]

ELSE [all current parameter values except the response type will be deleted from the database of NESS]

APPENDIX B

APPENDIX B

NESS PROGRAMMER'S MANUAL

FRAMES USED IN NESS

There are two types of frames used in NESS, static frames and dynamic frames. Static frames contain time invariant knowledge while dynamic frames created either by rule-bases or by the menu-input stage contain dynamic knowledge.

Static Frames

A list of all static frames used in NESS is available in the frame named "file_index" and slot named "frames." This frame is present in the file simxpert.l. Static frames store rule-bases, menu_input control knowledge, agendas, system specs, param specs and static data. Contents of each static frame can be found in the computer listing of NESS given in Appendix D. A brief description of the static data frames used in NESS follows. The three data frames used in NESS are: top_level_menu, output_display_menu and param_menu.

top_level_menu

This frame contains the top level menu of NESS i.e., the eight main functions of NESS. This frame looks as follows.

```
(top_level_menu
        (Exit to GENIE)
        (Set up initial parameter values)
        (Run simulation program)
        (Display current parameter values required for simulation)
        :
        :
        (Store current parameter values in a disk-file))
```

This frame is used by the menu_input frame top_level_control.

output_display_menu

This frame contains the menu of the outputs that can be generated by FORTRAN simulation program. This frame looks as follows.

```
(output_display_menu
        (Return to TOP LEVEL MENU)
        (Plot of omega)
        (Plot of theta)
        :
        (Numerical output generated by simulation))
```

This frame is used by the menu_input frame output_display_control.

param_menu

This frame contains a menu of all parameters known to NESS. It contains eighteen parameter names like TO, Tfinal, Kp matrix etc. This frame looks as follows.

```
(param_menu
        (Return to TOP LEVEL MENU)
        (Response type)
        (TO)
        (Tfinal)
        :
        :
        (Steady state error))
```

This frame is used by the menu_input frames disp_init_val_control and change_param_control.

Dynamic Frames

As mentioned earlier, dynamic frames store data obtained during an interactive session with NESS. The main purpose of these frames in NESS is to store values of parameters required by the simulation software. The frames used to store parameter values are initial_value, Kp, Kd, inertial_matrix, Quaternion, Y_matrix and response_chosen. Attributes about the parameters are stored in the frames Kp_matrix, Kd_matrix, inertial_mat and controller_type_desired. A listing of these frames created during a session with NESS can be found in Figure B.1. All the above mentioned frames are created by the value_input_rb rule-base.

initial_value

This frame stores values of parameters which are scalar in nature. It also stores some attributes concerning these parameters and some control knowledge required for an ordered firing of the rules in the backward-chained rule-base value_input_input_rb. The parameter values are stored in slots having the name of their respective parameters. The value of TO is stored in the slot TO, the value of deltaT is stored in the slot deltaT and so on. Attributes about parameters are stored in slots like frame_full, quaternion_initialized, init_theta=0_wanted, etc. Control knowledge for ordered firing of rules are stored in slots like first_step_found, second_step_found, tcontrol_1, ocontrol_1, etc. This frame is used by the disp_init_val_rb, run_rb, change_param_rb and user_rb rule-bases.

Kp Kd and inertial_matrix

These three frames are similar to each other, each containing the values of the controller or inertial matrix it is named after. Since all the three matrices contain nine elements, each of these frames contain nine slots, each slot corresponding to a unique element in the matrix. For example slot (1 2) corresponds to the element in the first row and second column of the matrix. These frames are used by the disp_init_val_rb and change_param_rb rule-bases.

```
(Kp ((1 2) (0.0))
    ((1 3) (0.0))
    ((2 1) (0.0))
    ((2 3) (0.0))
    ((3 1) (0.0))
    ((3 2) (0.0))
    ((1 1) (1))
    ((2 2) (1))
    ((3 3) (1)))
(Kd ((1 2) (0.0))
    ((1 3) (0.0))
    ((2 1) (0.0))
    ((2 3) (0.0))
    ((3 1) (0.0))
    ((3 2) (0.0))
    ((1 1) (1))
    ((2 2) (1))
    ((3 3) (1)))
(inertial_matrix ((1 1) (1))
                 ((3 3) (1))
                 ((2 2) (1))
                 ((1 2) (0.0))
                 ((1 3) (0.0))
                 ((2 1) (0.0))
                 ((2 3) (0.0))
                 ((3 1) (0.0))
                 ((3 2) (0.0)))
(Kp_matrix (matrix_type (diagonal))
           (equal_diagonal_terms (true))
           (matrix_full (true)))
(Kd_matrix (matrix_type (diagonal))
           (equal_diagonal_terms (true))
           (matrix_full (true)))
(inertial_mat (matrix_full (true)))
(controller_type_desired (proportional (yes))
                         (prop_cross_coup_bet_axes (no))
                         (same_proportional_control_for_all_axes (yes))
                         (derivative (yes))
                         (diff_cross_coup_bet_axes (no))
                         (same_differential_control_for_all_axes (yes)))
(response_chosen (response_type (!Frequency response!)))
(quaternion (wanted_by_user (no)))
(initial_value (quaternion_initialized (true))
               (integration_method (Euler))
               (axis_of_input_command (X))
               (T0 (0))
               (error (0.0))
               (init_theta=0_wanted (yes))
               (tcontrol_1 (true))
               (init_omega=0_wanted (yes))
               (ocontrol_1 (true))
               (Y_matrix_full (true))
               (first_step_found (true))
               (deltaT (0.01))
               (steady_state_error (2))
               (frame_full (true))
               (Tfinal (10.0))
               (amplitude (1))
               (init_frequency_value (0.0159154))
               (number_of_samples_per_period (256))
               (number_of_decades (3))
               (number_of_sampling_frequency_per_decade (3))
               (phase (0))
               (second_step_found (true)))
(Y_matrix (theta_x (0.0))
          (theta_y (0.0))
          (theta_z (0.0))
          (omega_x (0.0))
          (omega_y (0.0))
          (omega_z (0.0)))

;; An example of dynamic frames created by NESS.
```

Figure B.1

Kp_matrix Kd_matrix and inertial_mat

These frames contain attributes about the controller or inertial matrix they are named after. This is done by using slots like matrix_type, matrix_full, equal_diagonal_terms, etc.

controller_type_desired

This frame contains attributes about the type of controller desired by the user. This is done by using slots like proportional, derivative, etc.

response_chosen Quaternion Y_matrix

All these three frames store initial values. The first stores the type of response chosen by the user, the second stores the initial values of the Quaternion and the third stores initial values of the angular position and the angular velocity. These frames are used by the disp_init_val_rb and change_param_rb rule-bases.

In addition to these frames, four other frames are also created to store certain choices indicated by the user. They are top_level_choice, output_display_choice, disp_init_val_choice and change_param_choice.

top_level_choice

This frame stores the choice made by the user from the TOP LEVEL MENU of NESS. It is created by the top_level_agenda frame and is used by the stage_1_rb rule-base. It looks as follows.

```
(top_level_choice
      (top_level_menu
            (Run simulation program)))
```

output_display_choice

This frame stores the choice made by the user from menu of outputs generated by the simulation program. It is created by the output_display_agenda frame and is used by the output_display_rb rule-base. It looks as follows.

```
(output_display_choice
      (output_display_menu
            (Plot of omega)))
```

disp_init_val_choice

This frame stores the choice made by the user from the menu of parameters known to NESS, for the purpose of displaying the value of the chosen parameter. It is created by the disp_init_val_agenda frame and is used by the disp_init_val_rb rule-base. It looks as follows.

```
(disp_init_val_choice
        (param_menu
                (T0)))
```

change_param_choice

This frame stores the choice made by the user from the menu of parameters known to NESS, for the purpose of changing the value of the chosen parameter. It is created by the change_param_agenda frame and is used by the change_param_rb rule-base. It looks as follows.

```
(change_param_choice
        (param_menu
                (deltaT)))
```

Another frame called "user" is used in NESS to describe certain events and to store some attributes. This frame is created by the run_rb rule-base and is used by the stage_1_rb rule-base. It looks as follows.

```
(user
        (simulation_run
                (error_free))
        (likes_Tfinal_value
                (yes)))
```

DEMONS USED IN NESS

Demons are special purpose FRANZ LISP functions written to perform specific tasks incapable of being performed by GENIE. Demons are also used to perform tasks for which GENIE techniques might prove to be inefficient from the point of execution time, programming time and memory use. A brief description of the demons used in NESS follows.

Demons Present in File DEMON00.l

start_sim

This demon is used to run the FORTRAN-based simulation program in the FRANZ LISP environment.

store_values_in_lisparray_from_frame

This demon is used to store values in a FRANZ LISP array from a frame created by NESS.

printarray

This demon is used to display the values of an array or a matrix on the screen, with each row of the array displayed on one line on the screen.

load_eigen_values_in_frame

This demon calls a fortran function to calculate the eigenvalues of the system matrix. It then stores these eigenvalues in a frame. An heuristic about calculating Tfinal is also incorporated in this demon.

frequency_output_display

This demon is used to display some of the outputs generated by the simulation program in case a frequency response is simulated.

clear_display

This demon is used to clear the screen.

delim_display

This demon is used to print certain characters on the screen.

Demons Present in File DEMON01.1

setup_init_val_in_simula.inp

This demon is used to store in a file the parameter values in the format required by the simulation program. The simulation program gets its input data from this particular file.

Demons Present in File DEMON02.1

confirm

This demon is used by the matrix_change_rb rule-base to acquire information from the user.

change

This demon is used by the matrix_change_rb rule-base to change the value of an element of a matrix.

change_quaternion

This demon is used to change the initial values of the Quaternion.

Demons Present in File DEMON03.1

start_&_iterate

This demon is used to keep NESS in operation at its TOP LEVEL MENU until the user decides to exit from NESS by using option "Exit to GENIE" provided on the TOP LEVEL MENU.

loop

This demon is used to keep NESS at one of its secondary menu stages

i.e., the output_display_menu used to select a simulation output for display or the param_menu used to either display or change the values of parameters. The user can go to the TOP LEVEL MENU by using the option "Return to TOP LEVEL MENU" present on these secondary menus.

storage_of_initial_values

This demon is used to perform function 8 of the TOP LEVEL MENU i.e., "store current parameter values in a disk file."

change_Y_matrix

This demon is used to change the values of initial angular velocity and the initial angular position.

Demons Present in File DEMONAR.l

array_access_from_frame

This demon takes as arguments a name of an array or a matrix, a row number and a column number and returns a path to a frame containing elements of that particular array, down to the slot for that particular row number and column number.

store_values_in_frame_from_lispvector

This demon is used to store values from a FRANZ LISP vector to a frame created by NESS.

Demon Present in File LISPRC.l

ness

If you have exited to GENIE level after a session with NESS, running this demon from FRANZ LISP will enable you to come under the control of NESS without having to go back to the operating system level.

Coupling GENIE and the Simulation Software Using VAX 11/780 and FRANZ LISP

NESS is an expert system which utilizes the GENIE system as its inference engine. GENIE is documented elsewhere [GENIE Ref Manual] and is written in FRANZ LISP which is a LISP dialect designed for the UNIX operating system environment. FRANZ LISP provides a number of ways that a LISP process such as NESS can implement operating system calls. These calls allow NESS to do things like write a disk file containing the parameters that the simulation program needs, cause its execution and read the output files it creates as illustrated in Figure ____.

The most straightforward utilization of a system call is simply to include a FRANZ LISP function "exec" [Franz Manual CH 4] to cause the execution of a standard UNIX command, directly in a rule clause. For example output_display_rb_rule6 checks the precondition that insures that the simulation program has run and that the user wants to see a plot of theta which the simulation program would have deposited in a disk file

called "theta0plt.stp." The 'then' side of the rule looks as follows:

($then (exec cat theta0plt.stp)).

This causes the UNIX "cat" command to execute which simply copies the named file (theta0plt.stp) to the user's terminal.

A slightly more involved method is to write a demon (i.e., a special purpose LISP function) to perform some specific operation which may involve one or more calls to UNIX system functions. For example, a demon named "setup_init_val_in_simula.inp" calls the system functions "fileopen," "close," and "cprintf" [see Franz Manual] which does formatted file write operations. This demon is called by run_rb_rule1.

This rule also calls the demon "start_sim" which uses the FRANZ LISP function "process" [Franz Manual CH 6] to fork a child process which is the actual execution of the simulation program. The code for the rules and the demons is listed in Appendix D.

APPENDIX C

APPENDIX C
THEORETICAL BACKGROUND ON QUATERNION

1. INTRODUCTION

This appendix reviews the concepts and properties related to **Quaternion.** The definition and the algebra of Quaternion are covered in the following sections.

2. DEFINITION

In his classical book entitled Elements of Quaternions, Sir W. R. Hamilton defined Quaternion as

> "In fact it will be shown .... that there is an important sense in which we can conceive a scalar to be added to a vector; and that the sum so obtained, or the combination, 'Scalar plus Vector,' is a Quaternion." (Hamilton)

Throughout his book other definitions related to Quaternions can be found, where geometrical relations in the cartesian plane and space are involved. For example, Hamilton pointed out that

> "this essential connection of the complex relation between two lines, and which we have given the name of a geometrical quotient, with a system of four numerical elements, we have a motive for saying that the quotient of two vectors is generally a Quaternion."

Thus, we can expect from such a concept a variety of possible definitions for various applications. For example, a most widely used definition found in the aerospace literature states that

> "a Quaternion is a four-parameter system for uniquely specifying the attitude of a rigid body with respect to some reference frame." (Grubin)

The applicability of Quaternions to the solution of digital attitude control problems which involve transformations among different coordinate systems offers some advantages over the equivalent direction cosine solution (Ickes). As an example, let us consider the Space Telescope Pointing Control System (Glaese, et al.). The initial design was changed in 1974 to reduce program cost. They include: 1) moving the annular Support Systems Module components nearer to the composite center of mass; 2) elimination of the Image Motion Compensation system; 3) employment of four reaction wheels instead of single-gimbal control moment gyroscopes; and 4) performance of the emergency and backup functions by a magnetic torquer system. Significant cost reduction was obtained through the selection of reaction wheels as primary controllers and magnetic torquers as auxiliary actuators for momentum management and backup to the primary controllers in case of

failures. For overall system simplicity, strapdown integration is done in terms of the Quaternion, i.e. relative four parameter attitude variables. The advantages for using the Quaternion were summarized as:

1. a single algorithm for all attitude control modes;

2. small vehicle attitude errors at all times;

3. continuous updating of the strapdown calculation through a drift rate correction added to the reference frame rate; and

4. easy shaping of maneuver, scan or other rate profiles.

In the following, basic algebraic manipulations of Quaternions are described.

3. ALGEBRA OF QUATERNIONS

An extension of the previous definitions on the Quaternion may state that the Quaternion is a generalization of the set of complex numbers for the study of rotational motion (Glaese and Kennel). The mathematical expression for a Quaternion Q is given by

$$Q = [Q_1\ Q_2\ Q_3\ Q_4] = (\underline{Q}\ Q_4) = Q_4 + iQ_1 + jQ_2 + kQ_3 \qquad \text{(C-1)}$$

where $\underline{Q}$ represents the vector part or imaginary part of the Quaternion, Q4 the real or scalar component, and

$$i = \sqrt{-1},\ j = \sqrt{-1},\ k = \sqrt{-1} \qquad \text{(C-2)}$$

3.1 SUM OF QUATERNIONS

Let Q and R be Quaternions. Then, the sum of Q and R, called S, is given by:

$$S = Q + R = (\underline{Q}\ Q_4) + (\underline{R}\ R_4) = (\underline{S}\ S_4) \qquad \text{(C-3)}$$

where,

$$\underline{S} = \underline{Q} + \underline{R} = i(Q_1+R_1) + j(Q_2+R_2) + k(Q_3+R_3) \qquad \text{(C-4)}$$

and

$$S_4 = Q_4 + R_4 \qquad \text{(C-5)}$$

Therefore, the sum of Quaternions is commutative and associative.

3.2 PRODUCT OF QUATERNIONS

Because a vector and a real number are related to a Quaternion, the

following conditions are satisfied (Hamilton):

$$i*i = j*j = k*k = -1 \tag{C-6a}$$

and

$$i \times j = k,\ j \times k = i,\ k \times i = j \tag{C-6b}$$

The product of two Quaternions is then defined as follows:

$$P\ Q \circ R = (\underline{Q}\ Q_4) \circ (\underline{R}\ R_4) = (\underline{P}\ P_4) \tag{C-7}$$

where

$$\underline{P} = Q_4\underline{R} + R_4\underline{Q} + \underline{Q} \times \underline{R} \tag{C-8}$$

and

$$P_4 = Q_4\ R_4 - \underline{Q} \cdot \underline{R} \tag{C-9}$$

In the above equations, (.) and (x) represent the dot and cross product, respectively.

From equations (C-8) and (C-9) we observe that the (o) product is associative and distributive, but it is not commutative.

3.3 CONJUGATE OF QUATERNIONS

The conjugate of a Quaternion Q is given by:

$$Q = (iQ_1 + jQ_2 + kQ_3 + Q_4)^* = Q_4 - iQ_1 - jQ_2 - kQ_3 \tag{C-10}$$

3.4 INVERSE OF A QUATERNION

By combining definitions (C-7) and (C-10) we can define the inverse of a Quaternion. First let us consider

$$Q \circ Q^* = Q^* \circ Q = Q1^2 + Q2^2 + Q3^2 + Q4^2 \tag{C-11}$$

then, the inverse of the Quaternion Q, $Q^{(-1)}$, is given by:

$$Q_{-1} = \left[\frac{1}{(Q^* \circ Q)}\right] \cdot Q^* = \frac{1}{Q^* \circ Q} \tag{C-12}$$

3.5 TRIPLE PRODUCT OF QUATERNIONS

In order for us to study rotational motions in the three-dimensional space, we have to define the triple product of Quaternions, V'. The triple product is given by:

$$V' = Q^* \circ V \circ Q \tag{C-13}$$

This expression does not mix scalar and vector parts. For the vector part note from equation (C-13) that

$$(\underline{V}')^* = (Q^* \circ \underline{V} \circ Q)^* = Q^* \circ \underline{V}^* \circ Q = -Q^* \circ \underline{V} \circ Q = -\underline{V}' \qquad \text{(C-14)}$$

3.6 LENGTH OF VECTORS

From the previous results, we can find the length of a vector as follows:

$$\begin{aligned} |\underline{V}'|^2 &= \underline{V}' \cdot \underline{V}' = \underline{V}' \circ (\underline{V}')^* = \\ &= (Q^* \circ \underline{V} \circ Q) \circ (Q^* \circ \underline{V} \circ Q)^* = \\ &= (Q^* \circ \underline{V} \circ Q) \circ (Q^* \circ \underline{V}^* \circ Q) = \\ &= Q^* \circ \underline{V} \circ (Q \circ Q^*) \circ \underline{V}^* \circ Q = \\ &= (Q^* \circ Q)^2 \, \underline{V} \circ \underline{V}^* \end{aligned} \qquad \text{(C-15)}$$

It can be concluded that the length of a vector $\underline{V}$ is multiplied by the factor $(Q^* \circ Q)$, which is a real number. Thus, from equation (C-15) we notice that the length of a given vector $\underline{V}'$ in the prime coordinate frame is related to the norm of the vector $\underline{V}$ by

$$|\underline{V}'| = (Q \circ Q^*)^2 \, |\underline{V}|^2 \qquad \text{(C-16)}$$

4. VECTORIAL ALGEBRA OF QUATERNIONS

This section provides a closer look at the vectorial algebra of Quaternions.

4.1 NORMALIZATION

The real number $(Q^* \circ Q)$ is defined as the "common norm" (Hamilton). It can be assumed without loss of generality that

$$Q^* \circ Q = 1 \qquad \text{(C-17)}$$

and the concept of normalized Quaternions is introduced. Equation (C-1) is equivalent to the following expression:

$$Q = \cos(\emptyset/2) + \sin(\emptyset/2) \cdot \underline{u} \qquad \text{(C-18)}$$

where real and imaginary parts of the Quaternion are distinguished, and the vector $\underline{u}$ is an unitary vector (i.e., $\underline{u} \cdot \underline{u} = 1$). Equation (C-18) preserves the vector length.

4.2 ROTATION OPERATOR

From previous results, the rotation of a vector around another vector with some angle may be found using Quaternions. Substituting (C-18) into (C-13), we get

$$\underline{V}' = \text{COS}(\emptyset) \cdot \underline{V} + \text{SIN}(\emptyset) \cdot (\underline{u} \times \underline{V}) + [1-\text{COS}(\emptyset)](\underline{u} \cdot \underline{V})\underline{u} \qquad \text{(C-19)}$$

which is the general expression of a rotation of a vector $\underline{V}$ around the axis of the unitary vector $\underline{u}$, with an angle of rotation $\emptyset$.

Equation (C-19) is the same expression as the general rotation transformation

$$Rot(\vec{k},\phi) = \begin{bmatrix} k_x k_x \text{vers}\phi + \cos\phi & k_y k_x \text{vers}\phi - k_z \sin\phi & k_z k_x \text{vers}\phi + k_y \sin\phi & 0 \\ k_x k_y \text{vers}\phi + k_z \sin\phi & k_y k_y \text{vers}\phi + \cos\phi & k_z k_y \text{vers}\phi - k_x \sin\phi & 0 \\ k_x k_z \text{vers}\phi - k_y \sin\phi & k_y k_z \text{vers}\phi + k_x \sin\phi & k_z k_z \text{vers}\phi + \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(C-20)}$$

The vector $\underline{V}'$ is obtained by multiplying the matrix Rot($\underline{k}$,$\emptyset$) with $\underline{V}$. For details, see (Paul) or (Glaese & Kennel).

4.3 TIME-RATE CHANGE OF A VECTOR

The rate of change of the norm with respect to time is obtained by taking derivatives of equation (C-17):

$$d/dt(\dot{Q}^* \text{ o } Q) = (\dot{Q})^* \text{ o } Q + Q^* \text{ o } \dot{Q} = 0 \qquad \text{(C-21)}$$

From this equation it is shown that

$$\dot{Q}^* \text{ o } Q = - Q^* \text{ o } \dot{Q} = - (Q^* \text{ o } \dot{Q})^* \qquad \text{(C-22)}$$

which represents a particular vector. Let

$$Q^* \text{ o } \dot{Q} = 1/2 \ \underline{w} \qquad \text{(C-23)}$$

Premultiplying this equation by Q, we obtain

$$\dot{Q} = 1/2 \ Q \text{ o } \underline{w}$$

The evaluation of the rate of change of a vector with respect to time in two reference frames is performed in the following way:

$d/dt(\underline{V}) = d/dt(Q \text{ o } \underline{V}' \text{ o } Q^*) =$

$$= \dot{Q} \circ \underline{V}' \circ Q^* + Q \circ d/dt(\underline{V}' \circ Q^*) + Q \circ V' \circ \dot{Q}^* \qquad \text{(C-25)}$$

Substituting $d/dt(Q)$ and $d/dt(Q^*)$ obtained from equation (C-24) into equation (C-25), the rate of change of the vector $\underline{V}$ is

$$d/dt(\underline{V}) = Q \circ [d/dt(\underline{V}') + \underline{w} \times \underline{V}'] \circ Q^* \qquad \text{(C-26)}$$

In this equation $\underline{w}$ is identified as the relative angular velocity of the primed axes with respect to the unprimed (Glaese & Kennel).

The differential equation represented in equation (C-24) is known as the "Quaternion-rate equation" (Grubin). In the next section, a more detailed treatment of this equation and its implications are presented.

5. MATRIX ALGEBRA OF QUATERNIONS

The matrix representation of a Quaternion turns out to be a more convenient way of representing various Quaternion operations. For example, equation (C-7) defined the Quaternion product as

$$P = Q \circ R = \begin{pmatrix} Q_4\underline{R} + R_4\underline{Q} + \underline{Q} \times \underline{R} \\ Q_4R_4 - \underline{Q} \cdot \underline{R} \end{pmatrix} \qquad \text{(C-27)}$$

which can also be written as

$$P = Q \circ R = \begin{vmatrix} +Q_4 & -Q_3 & +Q_2 & +Q_1 \\ +Q_3 & +Q_4 & -Q_1 & +Q_2 \\ -Q_2 & +Q_1 & +Q_4 & +Q_3 \\ -Q_1 & -Q_2 & -Q_3 & +Q_4 \end{vmatrix} \begin{vmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{vmatrix} = \tilde{Q}\, R \qquad \text{(C-28)}$$

This representation clearly shows that the matrix $\tilde{Q}$ satisfy Quaternion properties and comprises "a matrix representation of Quaternion algebra with matrix multiplication corresponding to (o)" (Glaese & Kennel). The product in equation (C-28) can be also expressed in the alternate form:

$$P = \begin{vmatrix} +R_4 & +R_3 & -R_2 & +R_1 \\ -R_3 & +R_4 & +R_1 & +R_2 \\ +R_2 & -R_1 & +R_4 & +R_3 \\ -R_1 & -R_2 & -R_3 & +R_4 \end{vmatrix} \begin{vmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{vmatrix} = \overset{=}{R}\, Q \qquad \text{(C-29)}$$

Therefore we can show that:

$$P = \tilde{Q}\, R = \tilde{R}\, Q \tag{C-30}$$

$$D = A \circ B \circ C = (A \circ B) \circ C = (\widetilde{A \circ B})\, C = \tilde{A}\, \tilde{B}\, C = \tag{C-31}$$

$$= \tilde{C}\, \tilde{B}\, A = \tilde{A}\, \tilde{C}\, B = \tilde{C}\, \tilde{A}\, B$$

$$\text{=====} \quad \tilde{=}\ \tilde{=}$$

$$B \circ C = \tilde{C}\, \tilde{B} \tag{C-32}$$

Hence $\tilde{A}\, \tilde{C} = \tilde{C}\, \tilde{A}$, which is a very useful result.

5.1 QUATERNION-RATE EQUATIONS

In Section 4.3 the rate change of a vector $\underline{V}$ and the Quaternion-rate equations were derived. In this section, by applying some of the Quaternion matrix operations to them, more convenient representations will be derived.

First, let us consider equation (C-19) in product form:

$$\underline{V}' = Q^* \circ \underline{V} \circ Q = \tilde{Q}^*\, \underline{\tilde{V}}\, Q = \tilde{Q}^*\, \tilde{Q}\, \underline{V} \tag{C-33}$$

Vectors $\underline{V}'$ and $\underline{V}$ in the three-dimensional space are related through

$$\underline{V}' = M\, \underline{V} \tag{C-34}$$

where M is a direction cosine matrix and is a 3x3 matrix which transforms $\underline{V}$ coordinates to the primed coordinates $\underline{V}'$ by means of Q, i.e.,

$$M = Q^*\, \tilde{Q} \tag{C-35}$$

Referring now to equation (C-19) and substituting Ø by -Ø, we have

$$M = \underline{u}\, \underline{u}^T - \sin(Ø)u + \cos(Ø)[I - \underline{u}\, \underline{u}^T] \tag{C-36}$$

The matrix **u** is called the "cross product matrix" (Glaese & Kennel) and is formed by dropping the final row and final column of **u**. The identity matrix is represented by I and its dimension depends on the current implementation.

Solving equation (C-35), the expanded form of the direction cosine matrix is given in terms of the Quaternion parameters by (Glaese & Kennel):

$$M = \begin{vmatrix} Q_1^2 - Q_2^2 - Q_3^2 + Q_4^2 & 2(Q_1Q_2 + Q_3Q_4) & 2(Q_1Q_3 - Q_2Q_4) \\ 2(Q_2Q_1 - Q_3Q_4) & -Q_1^2 + Q_2^2 - Q_3^2 + Q_4^2 & 2(Q_2Q_3 + Q_1Q_4) \\ 2(Q_3Q_1 + Q_2Q_4) & 2(Q_3Q_2 - Q_1Q_4) & -Q_1^2 - Q_2^2 + Q_3^2 + Q_4^2 \end{vmatrix} \tag{C-37}$$

Finally, the Quaternion-rate equations are expressed in matrix form as

$$\dot{Q} = 1/2\ Q \circ \underline{w} = 1/2\ \underline{w}\ Q = 1/2 \left|\begin{array}{cccc||c} 0 & +w_3 & -w_2 & +w_1 & Q_1 \\ -w_3 & 0 & +w_1 & +w_2 & Q_2 \\ +w_2 & -w_1 & 0 & +w_3 & Q_3 \\ -w_1 & -w_2 & -w_3 & 0 & Q_4 \end{array}\right| \qquad \text{(C-38)}$$

or

$$\dot{Q} = 1/2\ Q \circ w = 1/2\ Q\ w = 1/2 \left|\begin{array}{ccc||c} +Q_4 & -Q_3 & +Q_2 & w_1 \\ +Q_3 & +Q_4 & -Q_1 & w_2 \\ -Q_2 & +Q_1 & +Q_4 & w_3 \\ -Q_1 & -Q_2 & -Q_3 & \end{array}\right| \qquad \text{(C-39)}$$

REFERENCES

Glaese, J.R., and Kennel, H.F., "Low Drag Attitude Control for SKYLAB Orbital Lifetime Extension," NASA Technical Memorandum TM-82412, pp. 37-52.

Glaese, J.R., et al., "Low-Cost Space Telescope Pointing Control System," Journal of Spacecraft, Vol. 13, No. 7, pp. 400-405, July, 1976.

Grubin, Carl, "Derivation of the Quaternion Scheme via the Euler Axis and Angle," Journal of Spacecraft, Vol. 7, No. 10, pp. 1261-1263, October, 1970.

Hamilton, Sir William, Elements of Quaternions, Longmans Green and Co., 1866.

Ickes, B.P., "A New Method for Performing Digital Control System Attitude Computations Using Quaternions," AIAA Journal, Vol. 8, No. 1, pp. 13-17, January, 1970.

Paul, R., Robot Manipulators: Mathematics, Programming, and Control, MIT Press, Cambridge, 1981.

APPENDIX D

NESS LISTING

```
(change_param_rb (if_removed (d_remove_rules))
                 (created_by (samir))
                 (created_on (12_25_85!))
                 (rules (change_param_rb_rule1)
                        (change_param_rb_rule4)
                        (change_param_rb_rule9)
                        (change_param_rb_rule10)
                        (change_param_rb_rule11)
                        (change_param_rb_rule12)
                        (change_param_rb_rule13)
                        (change_param_rb_rule17)
                        (change_param_rb_rule18)
                        (change_param_rb_rule3)
                        (change_param_rb_rule5)
                        (change_param_rb_rule14)
                        (change_param_rb_rule15)
                        (change_param_rb_rule16)
                        (change_param_rb_rule7)
                        (change_param_rb_rule6)
                        (change_param_rb_rule8)
                        (change_param_rb_rule19)
                        (change_param_rb_rule2))
                 (params_in_ifs
                  (change_param_choice
                   (param_menu
                    (rule (change_param_rb_rule1)
                          (change_param_rb_rule4)
                          (change_param_rb_rule9)
                          (change_param_rb_rule10)
                          (change_param_rb_rule11)
                          (change_param_rb_rule12)
                          (change_param_rb_rule13)
                          (change_param_rb_rule17)
                          (change_param_rb_rule18)
                          (change_param_rb_rule3)
                          (change_param_rb_rule5)
                          (change_param_rb_rule14)
                          (change_param_rb_rule15)
                          (change_param_rb_rule16)
                          (change_param_rb_rule7)
                          (change_param_rb_rule6)
                          (change_param_rb_rule8)
                          (change_param_rb_rule19)
                          (change_param_rb_rule2))))))
(change_param_rb_rule1 ($type (ifall))
                       ($if (triple (change_param_choice param_menu) = T0))
                       (created_by (samir))
                       (created_on (12_27_85!))
                       ($then (frem '(initial_value T0))
                              (delim_display)
                              (clause_print)
                              (clause_print
                               !Enter new value of starting time (T0):!)
                              (msg "  >> ")
                              (fput '(initial_value T0) (read)))
                       (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                       (modified_on (!6_14_85!)))
(change_param_rb_rule4 ($type (ifall))
                       ($if
                        (triple (change_param_choice param_menu)
                                =
```

```
                        !Multi-step integration error!))
                    (created_by (samir))
                    (created_on (!2_27_85!))
                    ($then (frem '(initial_value error))
                           (delim_display)
                           (clause_print)
                           (clause_print
                            !Enter new value of multi-step integration error:!)
                           (msg "  >> ")
                           (fput '(initial_value error) (read)))
                    (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                    (modified_on (!6_14_85!)))
(change_param_rb_rule9 ($type (ifall))
                    ($if
                     (triple (change_param_choice param_menu)
                             =
                             !Amplitude of input wave signal!))
                    (created_by (samir))
                    (created_on (!2_27_85!))
                    ($then (frem '(initial_value amplitude))
                           (delim_display)
                           (clause_print)
                           (clause_print
                            !Enter new value of amplitude of the input wave signal:!)
                           (msg "  >> ")
                           (fput '(initial_value amplitude) (read)))
                    (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                    (modified_on (!6_14_85!)))
(change_param_rb_rule10 ($type (ifall))
                     ($if
                      (triple (change_param_choice param_menu)
                              =
                              !Initial lowest frequency!))
                     (created_by (samir))
                     (created_on (!2_27_85!))
                     ($then (frem
                             '(initial_value init_frequency_value))
                            (delim_display)
                            (clause_print)
                            (clause_print
                             !Enter new value of initial lowest frequency:!)
                            (msg "  >> ")
                            (fput '(initial_value init_frequency_value)
                                  (read)))
                     (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                     (modified_on (!6_14_85!)))
(change_param_rb_rule11 ($type (ifall))
                     ($if
                      (triple (change_param_choice param_menu)
                              =
                              !Number of decades!))
                     (created_by (samir))
                     (created_on (!2_27_85!))
                     ($then (frem '(initial_value number_of_decades))
                            (delim_display)
                            (clause_print)
                            (clause_print
                             !Enter new value of number of decades:!)
                            (msg "  >> ")
                            (fput '(initial_value number_of_decades)
                                  (read)))
```

```
                        (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                        (modified_on (|6_14_85|)))
(change_param_rb_rule12 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu)
                                 =
                                 |Number of sampling frequencies/decade|))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($then (frem
                                '(initial_value
                                  number_of_sampling_frequency_per_decade))
                               (delim_display)
                               (clause_print)
                               (clause_print
                                |Enter new value of number of sampling frequencies per decade:|)
                               (msg "  >> ")
                               (fput '(initial_value
                                       number_of_sampling_frequency_per_decade)
                                     (read)))
                        (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                        (modified_on (|6_14_85|)))
(change_param_rb_rule13 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu)
                                 =
                                 |Phase of input wave signal|))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($then (frem '(initial_value phase))
                               (delim_display)
                               (clause_print)
                               (clause_print
                                |Enter new value of phase of the input wave signal:|)
                               (msg "  >> ")
                               (fput '(initial_value phase) (read)))
                        (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                        (modified_on (|6_14_85|)))
(change_param_rb_rule17 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu)
                                 =
                                 |Amplitude of impulse|))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($then (frem '(initial_value impulse_amplitude))
                               (delim_display)
                               (clause_print)
                               (clause_print
                                |Enter new value of amplitude of impulse:|)
                               (msg "  >> ")
                               (fput '(initial_value impulse_amplitude)
                                     (read)))
                        (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                        (modified_on (|6_14_85|)))
(change_param_rb_rule18 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu)
                                 =
                                 |Response type|))
                        (created_by (samir))
```

```
                        (created_on (!2_27_85!))
                        ($then (delim_display)
                               (clause_print)
                               (clause_print
                                !The response type must not be changed from this level, because of!)
                               (clause_print
                                !the lack of consistancy checking.  Please use option 2 of the TOP!)
                               (clause_print
                                !LEVEL MENU to change 'Response type'.!))
                        (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                        (modified_on (!6_14_85!)))
(change_param_rb_rule3 ($type (ifall))
                       ($if
                        (triple (change_param_choice param_menu)
                                =
                                !DeltaT (time increment)!))
                       (created_by (samir))
                       (created_on (!2_27_85!))
                       ($then (frem '(initial_value deltaT))
                              (delim_display)
                              (clause_print)
                              (clause_print
                               !Enter new value of time increment (deltaT):!)
                              (msg "  >> ")
                              (fput '(initial_value deltaT) (read)))
                       (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                       (modified_on (!6_14_85!)))
(change_param_rb_rule5 ($type (ifall))
                       ($if
                        (triple (change_param_choice param_menu)
                                =
                                !Steady state error!))
                       (created_by (samir))
                       (created_on (!2_27_85!))
                       ($then (frem '(initial_value steady_state_error))
                              (delim_display)
                              (clause_print)
                              (clause_print
                               !Enter new value of steady state error:!)
                              (msg "  >> ")
                              (fput '(initial_value steady_state_error)
                                    (read)))
                       (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                       (modified_on (!6_14_85!)))
(change_param_rb_rule14 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu) = !Kp Matrix!))
                        (created_by (samir))
                        (created_on (!2_27_85!))
                        ($then (delim_display)
                               (fput '(matrix_change matrix) 'Kp)
                               (fput '(matrix_change desired)
                                     'yes)
                               (reset_rule_base 'change_matrix_rb)
                               (forward 'change_matrix_rb))
                        (modified_by (!Juan J. Rodriguez & Bor-Jau Hsieh!))
                        (modified_on (!6_14_85!)))
(change_param_rb_rule15 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu) = !Kd Matrix!))
                        (created_by (samir))
```

```
                        (created_on (|2_27_85|))
                        ($then (delim_display)
                               (fput '(matrix_change matrix) 'Kd)
                               (fput '(matrix_change desired)
                                     'yes)
                               (reset_rule_base 'change_matrix_rb)
                               (forward 'change_matrix_rb))
                        (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                        (modified_on (|6_14_85|)))
(change_param_rb_rule16 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu)
                                 =
                                 |Inertial Matrix|))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($then (delim_display)
                               (fput '(matrix_change matrix)
                                     'inertial_matrix)
                               (fput '(matrix_change desired)
                                     'yes)
                               (reset_rule_base 'change_matrix_rb)
                               (forward 'change_matrix_rb))
                        (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                        (modified_on (|6_14_85|)))
(change_param_rb_rule7 ($type (ifall))
                       ($if
                        (triple (change_param_choice param_menu) = Quaternion))
                       (created_by (samir))
                       (created_on (|2_27_85|))
                       ($then (delim_display) (change_quaternion))
                       (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                       (modified_on (|6_14_85|)))
(change_param_rb_rule6 ($type (ifall))
                       ($if
                        (triple (change_param_choice param_menu)
                                =
                                |Initial values of Omega and Theta|))
                       (created_by (samir))
                       (created_on (|2_27_85|))
                       ($then (delim_display) (change_Y_matrix))
                       (modified_by (|Juan J. Rodriguez & Bor-Jau Hsieh|))
                       (modified_on (|6_14_85|)))
```

```
(change_param_rb_rule8 ($type (ifall))
                       (created_by (samir))
                       (created_on (12_27_85!))
                       ($if
                        (triple (change_param_choice param_menu)
                                =
                                !Frequency deltaT!))
                       ($then (frem
                               '(initial_value frequency_deltaT))
                              (delim_display)
                              (clause_print)
                              (clause_print Enter
                                            new
                                            value
                                            of
                                            Frequency
                                            deltaT.)
                              (clause_print


                               !Enter from the following values 256 or 512 or 1024!)
                              (msg "  >> ")
                              (fput '(initial_value
                                      frequency_deltaT)
                                    (read)))
                       (modified_by (samir))
                       (modified_on (18_1_85!)))
```

```
(change_param_rb_rule19 ($type (ifall))
                        ($if
                         (triple (change_param_choice param_menu)
                                 =
                                 !Axis of input command!))
                        (created_by (!Samir, Andy, and Juan!))
                        (created_on (!6_15_85!))
                        ($then (from
                                '(initial_value axis_of_input_command))
                               (delim_display)
                               (clause_print)
                               (clause_print
                                !Enter the new value for 'Axis of input command'!)
                               (clause_print
                                !The only values possible are X Y Z or none!)
                               (clause_print
                                !Enter X Y or Z (upper case only) or 'none' (lower case)!)
                               (msg "  >> ")
                               (fput '(initial_value axis_of_input_command)
                                     (read)))
                        (modified_by (samir))
                        (modified_on (!6_24_85!)))
(change_param_rb_rule2 ($type (ifall))
                       ($if (triple (change_param_choice param_menu) = Tfinal))
                       (created_by (samir))
                       (created_on (!2_27_85!))
                       ($then (clause_print
                               !You can change the value of Tfinal when you run the simulation program!)
                              (clause_print
                               !using option 3 of the TOP LEVEL MENU!)
                              (clause_print
                               !No change in value of Tfinal is possible now!))
                       (modified_by (samir))
                       (modified_on (!6_24_85!)))
(change_matrix_rb (if_removed (d_remove_rules))
                  (created_by (samir))
                  (created_on (!2_27_85!))
                  (rules (change_matrix_rb_rule1))
                  (params_in_ifs
                   (matrix_change (desired (rule (change_matrix_rb_rule1))))))
(change_matrix_rb_rule1 ($type (ifall))
                        ($if (triple (matrix_change desired) = yes) (confirm))
                        ($then (change)
                               (reset_rule_base 'change_matrix_rb))
                        ($else (from 'matrix_change))
                        (created_by (samir))
                        (created_on (!2_27_85!)))
```

```
(def start_sim
  (lambda nil
    (prog (ret_code)
          (setq ret_code (*process 'main))
          (cond ((neq ret_code 0)
                 (terpri)
                 (msg N "+-----------------------------------+")
                 (msg N "| Simulation terminated abnormally. |")
                 (msg N "+-----------------------------------+")
                 (return nil))
                (t (fput '(user simulation_run) 'error_free)
                   (fput '(user last_response_run)
                         (car
                          (fget_values
                           '(response_chosen response_type)))))))))
(def store_values_in_lisparray_from_frame
  (lambda (array_name)
    (prog (i)
          (setq i 0)
     loop1(prog (j)
                (setq j 0)
           loop2(store (array_name i j)
                       (times 1.0
                              (car
                               (fget_values
                                (array_access_from_frame array_name
                                                         (+ i 1)
                                                         (+ j 1))))))
                (setq j (+ j 1))
                (cond ((eq j 3) (return nil)))
                (go loop2))
          (setq i (+ i 1))
          (cond ((eq i 3) (return nil)))
          (go loop1)))))
(def printarray
  (lambda (array_name)
    (prog (i)
          (setq i 1)
     loop1(prog (j)
                (setq j 1)
           loop2(cprintf "%f  "
                         (times 1.0
                                (car
                                 (fget_values
                                  (array_access_from_frame array_name i j)))))
                (setq j (+ j 1))
                (cond ((eq j 4) (return nil)))
                (go loop2))
          (terpri)
          (setq i (+ i 1))
          (cond ((eq i 4) (return nil)))
          (go loop1)))))
(def load_eigen_values_in_frame
  (lambda nil
    (prog (ret_code1)
          (setq ret_code1
                (eigen (getd 'Kp)
                       (getd 'Kd)
                       (getd 'inertial_matrix)
                       (getd 'one_over_tau)
                       (getd 'eigr)
```

```
                (getd 'eigi)))
          (cond ((equal ret_code1 0.0)
                 (cond
                  ((equal (one_over_tau 0) 0.0)
                   (msg N "Uncontrolled system. Will not reach steady state.")
                   (msg N
                        "Probable cause: Kd or Kp matrix has all 0.0 elements.")
                   (msg N "Tfinal has been set to default value ")
                   (store (one_over_tau 0) 1.0)))
                 (store_values_in_frame_from_lispvector 'one_over_tau 1)
                 (store_values_in_frame_from_lispvector 'eigr 6)
                 (store_values_in_frame_from_lispvector 'eigi 6)
                 (msg N "Eigenvalues computation has been succesful.")
                 (terpri)
                 (cond
                  ((neq (car
                         (fget_values
                          '(response_chosen response_type)))
                        '|Step response|)
                   (print (quotient 1.0 (one_over_tau 0)))
                   (terpri)
                   (msg N "Given above is the calculated value of TAU.")
                   (terpri))))
                (t (msg N "Eigenvalues calculation has not been succesful.")
                   (msg N "Probable cause: the Inertial Matrix is singular.")
                   (msg N "Tfinal has been set to default value. ")
                   (store (one_over_tau 0) 1.0))))))
(def frequency_output_display
  (lambda (x)
    (prog (a)
          (msg N "Do you want to display for ")
          (print x)
          (msg N "     1) Amplitude.")
          (msg N "     2) Phase.")
     loop1(msg N "Please enter choice:")
          (msg N "   # ")
          (setq a (read))
          (cond ((eq a 1)
                 (cond ((eq x 'theta)
                        (exec cat theta0amp.frq)
                        (wait_a_while)
                        (clear_display))
                       (t (exec cat omega0amp.frq)
                          (wait_a_while)
                          (clear_display))))
                ((eq a 2)
                 (cond ((eq x 'theta)
                        (exec cat theta0pha.frq)
                        (wait_a_while)
                        (clear_display))
                       (t (exec cat omega0pha.frq)
                          (wait_a_while)
                          (clear_display))))
                (t (msg N "Your entry is not recognized.") (go loop1))))))
(def clear_display
  (lambda nil
    (princ (ascii 27))
    (princ (ascii 91))
    (princ (ascii 72))
    (princ (ascii 27))
    (princ (ascii 91))
```

```
    (princ (ascii 50))
    (princ (ascii 74))))
(def clear
  (lambda nil
    (princ (ascii 27))
    (princ (ascii 91))
    (princ (ascii 72))
    (princ (ascii 27))
    (princ (ascii 91))
    (princ (ascii 50))
    (princ (ascii 74))))
(def delim_display
  (lambda nil
    (msg (N 2) "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")))
(def wait_a_while
  (lambda nil
    (prog (i)
          (setq i 1)
     loop (cond ((lessp i 5000) (setq i (+ i 1)) (go loop)) (t (return))))))
```

```
;; Modified on Jun-15-85 (jjr)
(def setup_init_val_in_simula.inp
  (lambda nil
    (prog (x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 r q m dbin)
          (setq dbin (fileopen "simula.inp" "w"))
          (setq r (car (fget_values '(response_chosen response_type))))
          (setq q (car (fget_values '(quaternion wanted_by_user))))
          (setq m
                (car (fget_values '(initial_value integration_method))))
          (setq x1 'T)
          (setq x2 'F)
          (cond ((eq r '|Step response|)
                 (cprintf "%s " x1 dbin)
                 (cprintf "%s " x2 dbin)
                 (cprintf "%s " x2 dbin))
                ((eq r '|Frequency response|)
                 (cprintf "%s " x2 dbin)
                 (cprintf "%s " x1 dbin)
                 (cprintf "%s " x2 dbin))
                (t (cprintf "%s " x2 dbin)
                   (cprintf "%s " x2 dbin)
                   (cprintf "%s " x1 dbin)))
          (cond ((eq q 'yes) (cprintf "%s " x1 dbin))
                (t (cprintf "%s " x2 dbin)))
          (cond ((eq m 'Euler)
                 (cprintf "%s " x1 dbin)
                 (cprintf "%s " x2 dbin)
                 (cprintf "%s " x2 dbin))
                ((eq m '|Fourth-order Runge-Kutta|)
                 (cprintf "%s " x2 dbin)
                 (cprintf "%s " x1 dbin)
                 (cprintf "%s " x2 dbin))
                (t (cprintf "%s " x2 dbin)
                   (cprintf "%s " x2 dbin)
                   (cprintf "%s " x1 dbin)))
          (terpri dbin)
          (setq x1 (times 1.0 (car (fget_values '(initial_value T0)))))
          (setq x4 (times 1.0 (car (fget_values '(initial_value error)))))
          (setq x5 0.0)
          (setq x6 0.0)
          (setq x7 0.0)
          (setq x8 0.0)
          (setq x9 0.0)
          (setq x10 0.0)
          (setq x11 0.0)
          (setq x12 0.0)
          (setq x13 0.0)
          (setq x14 0.0)
          (setq x15 0.0)
          (cond ((eq r '|Step response|)
                 (setq x2
                       (times 1.0
                              (car
                               (fget_values '(initial_value Tfinal)))))
                 (setq x3
                       (times 1.0
                              (car
                               (fget_values '(initial_value deltaT)))))
                 (setq x6
                       (times 1.0
                              (car
```

```
                (fget_values
                 '(initial_value steady_state_error))))))
      ((eq r '|Impulse response|)
       (setq x2 (quotient 1.0 (one_over_tau 0)))
       (setq x3
             (times 1.0
                    (car
                     (fget_values '(initial_value deltaT)))))
       (setq x7
             (times 1.0
                    (car
                     (fget_values
                      '(initial_value impulse_amplitude))))))
      (t (setq x2 (quotient 1.0 (one_over_tau 0)))
         (setq x3
               (times 1.0
                      (car
                       (fget_values
                        '(initial_value
                          frequency_deltaT)))))
         (setq x8
               (times 1.0
                      (car
                       (fget_values
                        '(initial_value amplitude)))))
         (setq x9
               (times 1.0
                      (car
                       (fget_values
                        '(initial_value init_frequency_value)))))
         (setq x11
               (times 1.0
                      (car
                       (fget_values
                        '(initial_value number_of_decades)))))
         (setq x10
               (times 1.0
                      (car
                       (fget_values
                        '(initial_value phase)))))
         (setq x12
               (times 1.0
                      (car
                       (fget_values
                        '(initial_value
                          number_of_sampling_frequency_per_decade)))))))
(cprintf "%f " x1 dbin)
(cprintf "%f " x2 dbin)
(cprintf "%f " x3 dbin)
(cprintf "%f " x4 dbin)
(cprintf "%f " x5 dbin)
(terpri dbin)
(cprintf "%f " x6 dbin)
(cprintf "%f " x7 dbin)
(cprintf "%f " x8 dbin)
(cprintf "%f " x9 dbin)
(cprintf "%f " x10 dbin)
(terpri dbin)
(cprintf "%f " x11 dbin)
(cprintf "%f " x12 dbin)
(cond ((eq
```

```
                (car
                    (fget_values '(initial_value axis_of_input_command)))
                'X) (setq x13 1.0))
            ((eq
                (car
                    (fget_values '(initial_value axis_of_input_command)))
                'Y) (setq x14 1.0))
            ((eq
                (car
                    (fget_values '(initial_value axis_of_input_command)))
                'Z) (setq x15 1.0)))
        (cprintf "%f " x13 dbin)
        (cprintf "%f " x14 dbin)
        (cprintf "%f " x15 dbin)
        (terpri dbin)
        (setq x1 (times 1.0 (car (fget_values '(Y_matrix omega_x)))))
        (setq x2 (times 1.0 (car (fget_values '(Y_matrix omega_y)))))
        (setq x3 (times 1.0 (car (fget_values '(Y_matrix omega_z)))))
        (setq x4 (times 1.0 (car (fget_values '(Y_matrix theta_x)))))
        (setq x5 (times 1.0 (car (fget_values '(Y_matrix theta_y)))))
        (setq x6 (times 1.0 (car (fget_values '(Y_matrix theta_z)))))
        (setq x7 0.0)
        (setq x8 0.0)
        (setq x9 0.0)
        (setq x10 0.0)
        (cond
         ((eq q 'yes)
          (setq x7
                (times 1.0
                       (car (fget_values '(quaternion roll_angle)))))
          (setq x8
                (times 1.0
                       (car (fget_values '(quaternion yaw_angle)))))
          (setq x9
                (times 1.0
                       (car (fget_values '(quaternion pitch_angle)))))))
        (cprintf "%f " x1 dbin)
        (cprintf "%f " x2 dbin)
        (cprintf "%f " x3 dbin)
        (cprintf "%f " x4 dbin)
        (cprintf "%f " x5 dbin)
        (terpri dbin)
        (cprintf "%f " x6 dbin)
        (cprintf "%f " x7 dbin)
        (cprintf "%f " x8 dbin)
        (cprintf "%f " x9 dbin)
        (cprintf "%f " x10 dbin)
        (terpri dbin)
        (prog nil
              (setq x6 1)
         loop1(cond ((eq x6 1) (setq x9 'Kp))
                    ((eq x6 2) (setq x9 'Kd))
                    (t (setq x9 'inertial_matrix)))
              (prog nil
                    (setq x7 1)
               loop2(prog nil
                          (setq x8 1)
                     loop3(setq x1
                                (times 1.0
                                       (car
                                        (fget_values
```

```
                                (array_access_from_frame x9
                                                         x7
                                                         x8)))))
                        (cprintf "%f " x1 dbin)
                        (setq x8 (+ x8 1))
                        (cond ((eq x8 4) (return nil)))
                        (go loop3))
                    (setq x7 (+ x7 1))
                    (cond ((eq x7 4) (return nil)))
                    (go loop2))
               (terpri dbin)
               (setq x6 (+ x6 1))
               (cond ((eq x6 4) (return nil)))
               (go loop1))
          (close dbin)
          (exec /etc/unixtovms simula.inp))))
```

```
(def confirm
  (lambda nil
    (prog (ans)
          (terpri)
          (fprint '(matrix_change matrix))
          (msg N "Do you want to alter any elements of the above matrix ?" N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
     loop1(msg N "Please enter choice:")
          (msg N "   # ")
          (setq ans (read))
          (cond ((eq ans 1) (return '(1 (5 5))))
                ((eq ans 2) (return '(-1 (5 5))))
                (t (msg N "Your entry is not recognized. Try again.")
                   (go loop1))))))
(def change
  (lambda nil
    (prog (row col val mat)
          (setq mat (car (fget_values '(matrix_change matrix))))
          (msg N "About the element you want to alter. Please,")
          (msg N "- Enter the row number = ")
          (setq row (read))
          (msg N "- Enter the col number = ")
          (setq col (read))
          (msg N "- Enter the new  value = ")
          (setq val (read))
          (frem (array_access_from_frame mat row col))
          (fput (array_access_from_frame mat row col) val))))
(def change_quaternion
  (lambda nil
    (prog (a b c d)
          (msg N
               "Do you want the Quaternion block to be included in simulation ?"
               N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
     loop1(msg N "Please enter choice:")
          (msg N "   # ")
          (setq a (read))
          (cond ((eq a 1)
                 (prog nil
                       (frem '(quaternion wanted_by_user))
                       (fput '(quaternion wanted_by_user) 'yes)
                       (msg N "Do you want a new value for Roll angle ?")
                       (msg N
                            "Be sure you had earlier entered a value for Roll angle."
                            N)
                       (msg N "     1) yes")
                       (msg N "     2) no" N)
                  loop2(msg N "Please enter choice:")
                       (msg N "   # ")
                       (setq b (read))
                       (cond ((eq b 1)
                              (frem '(quaternion roll_angle))
                              (msg N "Enter the new value of initial angle: ")
                              (fput '(quaternion roll_angle) (read)))
                             ((eq b 2) (msg N "No change to Roll angle."))
                             (t (msg N
                                     "Your entry is not recognized. Try again.")
                                (go loop2)))
                       (msg N "Do you want a new value for Pitch angle ?")
```

```
      (msg N
           "Be sure you had earier entered a value for Pitch angle."
           N)
      (msg N "     1) yes")
      (msg N "     2) no" N)
 loop3(msg N "Please enter choice:")
      (msg N "   # ")
      (setq c (read))
      (cond ((eq c 1)
             (frem '(quaternion pitch_angle))
             (msg N "Enter the new value of initial angle: ")
             (fput '(quaternion pitch_angle) (read)))
            ((eq c 2) (msg N "No change to Pitch angle."))
            (t (msg N
                    "Your entry is not recognized. Try again.")
               (go loop3)))
      (msg N "Do you want a new value for Yaw angle ?")
      (msg N
           "Be sure you had earier entered a value for Yaw angle."
           N)
      (msg N "     1) yes")
      (msg N "     2) no" N)
 loop4(msg N "Please enter choice:")
      (msg N "   # ")
      (setq d (read))
      (cond ((eq d 1)
             (frem '(quaternion yaw_angle))
             (msg N "Enter the new value of initial angle: ")
             (fput '(quaternion yaw_angle) (read)))
            ((eq d 2) (msg N "No change to Yaw angle."))
            (t (msg N
                    "Your entry is not recognized. Try again.")
               (go loop4)))))
((eq a 2)
 (msg N
      "The simulation program will not use the Quaternion block.")
 (msg N "That is, you will be running PROTOTYPE SYSTEM 0." N)
 (frem '(quaternion wanted_by_user))
 (fput '(quaternion wanted_by_user) 'no))
(t (msg N "Your entry is not recognized. Try again.")
   (go loop1))))))
```

```
;; Modified by Juan J. Rodriguez-Moscoso &
;;              Bor-Jau Hsieh    (16-Jun-85)
(def start_&_iterate
  (lambda nil
    (prog nil
     loop (eval_agenda 'top_level_agenda)
          (cond
           ((eq (car (fget_values '(top_level_choice top_level_menu)))
                '|Exit to GENIE|)
            (clear_display)
            (return (genie))))
          (terpri)
          (terpri)
          (wait_for_user)
          (terpri)
          (terpri)
          (go loop))))

(def loop
  (lambda (option)
    (prog nil
     loop1(cond ((eq option 'disp)
                 (cond ((eq (car
                              (fget_values
                               '(disp_init_val_choice param_menu)))
                            '|Return to TOP LEVEL MENU|)
                        (return nil))
                       (t (wait_for_user)
                          (clear_display)
                          (eval_agenda 'disp_init_val_agenda))))
                ((eq option 'change)
                 (cond ((eq (car
                              (fget_values
                               '(change_param_choice param_menu)))
                            '|Return to TOP LEVEL MENU|)
                        (return nil))
                       (t (wait_for_user)
                          (clear_display)
                          (eval_agenda 'change_param_agenda))))
                ((eq option 'out_disp)
                 (cond ((eq (car
                              (fget_values
                               '(output_display_choice output_display_menu)))
                            '|Return to TOP LEVEL MENU|)
                        (return nil))
                       (t (wait_for_user)
                          (clear_display)
                          (eval_agenda 'output_display_agenda)))))
          (go loop1))))

(def storage_of_initial_values
  (lambda nil
    (store_frames_in_file '(Kp Kd
                            inertial_matrix
                            Kp_matrix
                            Kd_matrix
                            inertial_mat
                            controller_type_desired
                            response_chosen
                            quaternion
                            initial_value
```

```
                        Y_matrix)
                      (read))))

(def change_Y_matrix
  (lambda nil
    (prog (a)
          (msg N "Do you want to change the value of Theta(x) ?" N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
     loop1(msg N "Please enter choice:")
          (msg N "   # ")
          (setq a (read))
          (cond ((eq a 1)
                 (frem '(Y_matrix theta_x))
                 (msg N "Enter the new value for Theta(x) = ")
                 (fput '(Y_matrix theta_x) (read)))
                ((eq a 2) (msg N "No change to Theta(x)."))
                (t (msg N "Your entry is not recognized. Try again.")
                   (go loop1)))
          (msg N "Do you want to change the value of Theta(y) ?" N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
     loop2(msg N "Please enter choice:")
          (msg N "   # ")
          (setq a (read))
          (cond ((eq a 1)
                 (frem '(Y_matrix theta_y))
                 (msg N "Enter the new value for Theta(y) = ")
                 (fput '(Y_matrix theta_y) (read)))
                ((eq a 2) (msg N "No change to Theta(y)."))
                (t (msg N "Your entry is not recognized. Try again.")
                   (go loop2)))
          (msg N "Do you want to change the value of Theta(z) ?" N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
     loop3(msg N "Please enter choice:")
          (msg N "   # ")
          (setq a (read))
          (cond ((eq a 1)
                 (frem '(Y_matrix theta_z))
                 (msg N "Enter the new value for Theta(z) = ")
                 (fput '(Y_matrix theta_z) (read)))
                ((eq a 2) (msg N "No change to Theta(z)."))
                (t (msg N "Your entry is not recognized. Try again.")
                   (go loop3)))
          (msg N "Do you want to change the value of Omega(x) ?" N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
     loop4(msg N "Please enter choice:")
          (msg N "   # ")
          (setq a (read))
          (cond ((eq a 1)
                 (frem '(Y_matrix omega_x))
                 (msg N "Enter the new value for Omega(x) = ")
                 (fput '(Y_matrix omega_x) (read)))
                ((eq a 2) (msg N "No change to Omega(x)."))
                (t (msg N "Your entry is not recognized. Try again.")
                   (go loop4)))
          (msg N "Do you want to change the value of Omega(y) ?" N)
          (msg N "     1) yes")
          (msg N "     2) no" N)
```

```
loop5(msg N "Please enter choice:")
     (msg N "   # ")
     (setq a (read))
     (cond ((eq a 1)
            (from '(Y_matrix omega_y))
            (msg N "Enter the new value for Omega(y) = ")
            (fput '(Y_matrix omega_y) (read)))
           ((eq a 2) (msg N "No change to Omega(y)."))
           (t (msg N "Your entry is not recognized. Try again.")
              (go loop5)))
     (msg N "Do you want to change the value of Omega(z) ?" N)
     (msg N "      1) yes")
     (msg N "      2) no" N)
loop6(msg N "Please enter choice:")
     (msg N "   # ")
     (setq a (read))
     (cond ((eq a 1)
            (from '(Y_matrix omega_z))
            (msg N "Enter the new value for Omega(z) = ")
            (fput '(Y_matrix omega_z) (read)))
           ((eq a 2) (msg N "No change to Omega(z)."))
           (t (msg N "Your entry is not recognized. Try again.")
              (go loop6))))))
```

```
(def array_access_from_frame
  (lambda (array_name row_num col_num)
    (list array_name (list row_num col_num))))

(def define_array
  (lambda (array_name row_bnd col_bnd)
    (array array_name flonum row_bnd col_bnd)))

(def store_values_in_frame_from_lispvector
  (lambda (array_name row_bnd)
    (prog (i)
          (setq i 1)
     loop1(prog (j)
                (setq j 1)
           loop2(frem (array_access_from_frame array_name i j))
                (fput (array_access_from_frame array_name i j)
                      (array_name (- i 1)))
                (setq j (+ j 1))
                (cond ((eq j 2) (return nil)))
                (go loop2))
          (setq i (+ i 1))
          (cond ((eq i (+ row_bnd 1)) (return nil)))
          (go loop1))))

(def form_identity_matrix
  (lambda (array_name row_bnd col_bnd)
    (prog (i)
          (setq i 1)
     loop1(prog (j)
                (setq j 1)
           loop2(cond ((eq i j) (store (array_name i j) 1.0))
                      (t (store (array_name i j) 0.0)))
                (setq j (+ j 1))
                (cond ((eq j (+ col_bnd 1)) (return nil)))
                (go loop2))
          (setq i (+ i 1))
          (cond ((eq i (+ row_bnd 1)) (return nil)))
          (go loop1))))
```

```
;;  Initialization program for running either:
;;      1) NASA Simulation Expert System,
;;      2) GENIE, or
;;      3) Franz Lisp Opus 38.79
;;** Written by
;;      Bor-Jau Hsieh (Andy) and
;;      Juan J. Rodriguez-Moscoso (May-11-85)
;;  Revised by
;;      Juan J. Rodriguez-Moscoso and
;;      Bor-Jau Hsieh (Andy)      (Jun-09-85)

;; NEES means NASA Expert Simulation System required by user in LISP
;;(defun ness ()
;;      (cond ((equal ans 1)(eval_agenda 'start_agenda))
;;            (t (msg N "You should use option 1 to start loading NESS" N )
;;               (startup))))
(defun ness()
      (start_&_iterate))

;; NESS_LOAD means NASA Expert Simulation System at "load" time.
(def ness_load
  (lambda nil
        (prog (answer1 answer2)
        (clear_display)
        (heading)
        (terpri)
        (msg N "Loading GENIE, and NESS.... (please be patient)" N)
        (include /u/gait/henrik/genie))
        (set_up_frames_from_file 'simxpert.l)
        (clear_display)
        (eval_agenda 'start_agenda)
        (clear_display]

;; HEADING stands for the heading of the NESS
(defun heading()
      (msg N "+=-===========================================================+")
      (msg N "!       Welcome to NASA Expert Simulation System (NESS)      !")
      (msg N "+=============================================================+")
]
;; STARTUP stands for starting up the Lisp environment
(defun startup()
      (prog (answer)
            (terpri)(terpri)
            (msg N "Do you want to run:")
            (terpri)
            (msg N "   1) NASA Expert Simulation System ?")
            (msg N "   2) GENIE (GENeric Inference Engine) ?")
            (msg N "   3) Franz Lisp ?")
            (terpri)
            loop
            (msg N "Please enter choice(s):")
            (msg N "    # ")
            (setq answer (read))
            (cond ((equal answer 1)
                   (clear_display)
                   (prog (ans)
                   (msg N "Before running the NASA Expert Simulation System,")
                   (msg N "did you type in 'SHELL' while you were under VMS?")
                   (terpri)
                   (msg N "    1) yes")
                   (msg N "    2) no ")
```

```
                    (msg N "    3) don't know")
                    loop1
                    (terpri)
                    (msg N "Please enter choice(s):")
                    (msg N "   # ")
                    (setq ans (read))
                    (cond ((equal ans 1)(return (ness_load)))
                          ((equal ans 2)
                           (msg N "You are being sent to the Operating ")
                           (princ "System you started in.")
                           (msg N "Please type 'SHELL' now if you are ")
                           (princ "under VMS O/S.")
                           (terpri)
                           (return (exit)))
                          ((equal ans 3)
                           (msg N "You are being sent back to the ")
                           (princ "Operating System you were before")
                           (msg N "you typed in 'lisp'. If you are ")
                           (princ "under VMS, type 'SHELL' to enter")
                           (msg N "into EUNICE. If not, just type 'lisp'")
                           (princ " again.")
                           (terpri)
                           (return (exit)))
                          (t (terpri)
                             (msg N "Please enter 1, 2 or 3 only")
                             (go loop1)
                          )
                    )
                    )
                   )
             )
             (cond ((equal answer 2)
                    (clear_display)
                    (msg N "Loading GENIE.... " N )
                    (include /u/gait/henrik/genie)
                    (set_up_frames_from_file 'simxpert.1)
                    (return (genie))))
             (cond ((equal answer 3)(return nil)))
             (cond ((equal answer 2)(return nil)))
             (cond ((equal answer 1)(return nil))
                   (t
                    (terpri)
                    (msg N "Please enter 1, 2 or 3 only")
                    (go loop]

;; clear_display stands for clearing the display on VT100 type terminal.
(defun clear_display()
             (princ (ascii 27))
             (princ (ascii 91))
             (princ (ascii 72))
             (princ (ascii 27))
             (princ (ascii 91))
             (princ (ascii 50))
             (princ (ascii 74]

;; Running 'startup' function
(clear_display)
(startup)
```

```
(value_input_rb (params_in_ifs (initial_value (equal_theta_values_for_all_axes
                                                (rule (value_input_rb_rule22)
                                                      (value_input_rb_rule23)))
                                               (equal_omega_values_for_all_axes
                                                (rule (value_input_rb_rule26)
                                                      (value_input_rb_rule27)))
                                               (first_step_found
                                                (rule (value_input_rb_rule3)))
                                               (deltaT
                                                (rule (value_input_rb_rule3)))
                                               (steady_state_error
                                                (rule (value_input_rb_rule3)))
                                               (second_step_found
                                                (rule (value_input_rb_rule3)))
                                               (imp_amp_found
                                                (rule (value_input_rb_rule3)))
                                               (init_theta=0_wanted
                                                (rule (value_input_rb_rule21)))
                                               (integration_method
                                                (rule (value_input_rb_rule29)
                                                      (value_input_rb_rule2)))
                                               (tcontrol_1
                                                (rule (value_input_rb_rule24)))
                                               (tcontrol_2
                                                (rule (value_input_rb_rule24)))
                                               (tcontrol_3
                                                (rule (value_input_rb_rule24)))
                                               (ocontrol_1
                                                (rule (value_input_rb_rule24)))
                                               (ocontrol_2
                                                (rule (value_input_rb_rule24)))
                                               (ocontrol_3
                                                (rule (value_input_rb_rule24)))
                                               (quaternion_initialized
                                                (rule (value_input_rb_rule2)))
                                               (axis_of_input_command
                                                (rule (value_input_rb_rule2)))
                                               (T0
                                                (rule (value_input_rb_rule2)))
                                               (error
                                                (rule (value_input_rb_rule2)))
                                               (Y_matrix_full
                                                (rule (value_input_rb_rule2)))
                                               (init_omega=0_wanted
                                                (rule (value_input_rb_rule25)))
                                               (frame_full
                                                (rule (value_input_rb_rule1)))
                                               (amplitude
                                                (rule (value_input_rb_rule4)))
                                               (init_frequency_value
                                                (rule (value_input_rb_rule4)))
                                               (frequency_deltaT
                                                (rule (value_input_rb_rule4)))
                                               (number_of_decades
                                                (rule (value_input_rb_rule4)))
                                               (number_of_sampling_frequency_per_decade
                                                (rule (value_input_rb_rule4)))
                                               (phase
                                                (rule (value_input_rb_rule4))))
                                  (Kp_matrix (matrix_symmetric
                                              (rule (value_input_rb_rule14)
```

```
                                        (value_input_rb_rule16)))
                                  (matrix_type
                                   (rule (value_input_rb_rule18)))
                                  (equal_diagonal_terms
                                   (rule (value_input_rb_rule18)))
                                  (matrix_full
                                   (rule (value_input_rb_rule2))))
                           (Kd_matrix (matrix_symmetric
                                   (rule (value_input_rb_rule17)
                                         (value_input_rb_rule15)))
                                  (matrix_type
                                   (rule (value_input_rb_rule19)))
                                  (equal_diagonal_terms
                                   (rule (value_input_rb_rule19)))
                                  (matrix_full
                                   (rule (value_input_rb_rule2))))
                           (controller_type_desired (same_differential_control_for_all_axes
                                                     (rule (value_input_rb_rule10)
                                                           (value_input_rb_rule12)))
                                                    (derivative
                                                     (rule
                                                      (value_input_rb_rule5)))
                                                    (proportional
                                                     (rule
                                                      (value_input_rb_rule6)))
                                                    (same_proportional_control_for_all_axes
                                                     (rule (value_input_rb_rule9)
                                                           (value_input_rb_rule11)))

; <<<<< start back on the left <<<<<
    (diff_cross_coup_bet_axes
     (rule (value_input_rb_rule8)
           (value_input_rb_rule10)
           (value_input_rb_rule17)
           (value_input_rb_rule12)
           (value_input_rb_rule15)))
; >>>>> continue on the right >>>>>

; <<<<< start back on the left <<<<<
    (prop_cross_coup_bet_axes
     (rule (value_input_rb_rule7)
           (value_input_rb_rule9)
           (value_input_rb_rule11)
           (value_input_rb_rule14)
           (value_input_rb_rule16)))
; >>>>> continue on the right >>>>>
)
                           (Kd ((1 1)
                                (rule (value_input_rb_rule10)
                                      (value_input_rb_rule17)
                                      (value_input_rb_rule12)
                                      (value_input_rb_rule15)))
                               ((2 2)
                                (rule (value_input_rb_rule17)
                                      (value_input_rb_rule12)
                                      (value_input_rb_rule15)))
                               ((3 3)
                                (rule (value_input_rb_rule17)
                                      (value_input_rb_rule12)
                                      (value_input_rb_rule15)))
```

```
        ((1 2)
         (rule (value_input_rb_rule17)
               (value_input_rb_rule15)))
        ((1 3)
         (rule (value_input_rb_rule17)
               (value_input_rb_rule15)))
        ((2 3)
         (rule (value_input_rb_rule17)
               (value_input_rb_rule15)))
        ((2 1) (rule (value_input_rb_rule17)))
        ((3 1) (rule (value_input_rb_rule17)))
        ((3 2) (rule (value_input_rb_rule17))))
    (Kp ((1 1)
         (rule (value_input_rb_rule9)
               (value_input_rb_rule11)
               (value_input_rb_rule14)
               (value_input_rb_rule16)))
        ((2 2)
         (rule (value_input_rb_rule11)
               (value_input_rb_rule14)
               (value_input_rb_rule16)))
        ((3 3)
         (rule (value_input_rb_rule11)
               (value_input_rb_rule14)
               (value_input_rb_rule16)))
        ((1 2)
         (rule (value_input_rb_rule14)
               (value_input_rb_rule16)))
        ((1 3)
         (rule (value_input_rb_rule14)
               (value_input_rb_rule16)))
        ((2 3)
         (rule (value_input_rb_rule14)
               (value_input_rb_rule16)))
        ((2 1) (rule (value_input_rb_rule16)))
        ((3 1) (rule (value_input_rb_rule16)))
        ((3 2) (rule (value_input_rb_rule16))))
    (response_chosen
     (response_type
      (rule (value_input_rb_rule3)
            (value_input_rb_rule2))))
    (Y_matrix (theta_x
               (rule (value_input_rb_rule22)
                     (value_input_rb_rule23)))
              (omega_x
               (rule (value_input_rb_rule26)
                     (value_input_rb_rule27)))
              (theta_y
               (rule (value_input_rb_rule23)))
              (theta_z
               (rule (value_input_rb_rule23)))
              (omega_y
               (rule (value_input_rb_rule27)))
              (omega_z
               (rule (value_input_rb_rule27))))
    (quaternion (wanted_by_user
                 (rule (value_input_rb_rule20)))
                (roll_angle
                 (rule (value_input_rb_rule20)))
                (pitch_angle
                 (rule (value_input_rb_rule20)))
```

```
                                (yaw_angle
                                 (rule (value_input_rb_rule20))))
                          (inertial_mat
                           (matrix_full (rule (value_input_rb_rule2))))
                          (inertial_matrix
                           ((1 1) (rule (value_input_rb_rule13)))))
        (if_removed (d_remove_rules))
        (created_by (|Samir, Juan, and Andy|))
        (created_on (|6_13_85|))
        (rules (value_input_rb_rule3)
               (value_input_rb_rule5)
               (value_input_rb_rule6)
               (value_input_rb_rule18)
               (value_input_rb_rule19)
               (value_input_rb_rule21)
               (value_input_rb_rule22)
               (value_input_rb_rule23)
               (value_input_rb_rule26)
               (value_input_rb_rule27)
               (value_input_rb_rule29)
               (value_input_rb_rule20)
               (value_input_rb_rule24)
               (value_input_rb_rule8)
               (value_input_rb_rule10)
               (value_input_rb_rule17)
               (value_input_rb_rule12)
               (value_input_rb_rule2)
               (value_input_rb_rule25)
               (value_input_rb_rule1)
               (value_input_rb_rule7)
               (value_input_rb_rule9)
               (value_input_rb_rule11)
               (value_input_rb_rule14)
               (value_input_rb_rule16)
               (value_input_rb_rule4)
               (value_input_rb_rule13)
               (value_input_rb_rule15))
        (param_specs (frame_full (value_type (literal))
                                 (param_cf (no))
                                 (find_strategy (try_rules) (ask)))
                     (mode (value_type (literal))
                           (value_multiplicity (single))
                           (value_required (yes))
                           (param_cf (no))
                           (legal_values (step) (frequency)))
                     (tcontrol_1 (find_strategy (try_rules)))
                     (tcontrol_2 (find_strategy (try_rules)))
                     (ocontrol_1 (find_strategy (try_rules)))
                     (ocontrol_2 (find_strategy (try_rules)))
                     (imp_amp_ok (value_type (literal))
                                 (value_multiplicity (single))
                                 (value_required (yes))
                                 (legal_values (yes) (no))
                                 (prompt_format (menu_input))
                                 (message_format (print
                                                  |Do you want the calculated value of 'Impulse amplitude' ?|)
                                                 (print
                                                  |If you say NO, then you can give your own value.|)))
                     (axis_of_input_command (value_multiplicity
                                             (single))
                                            (value_required (yes))
```

```
                                                (prompt_format
                                                 (menu_input))
                                                (legal_values (X)
                                                              (Y)
                                                              (Z)
                                                              (none))

; <<<<< start back on the left <<<<<
    (message_format (print
                     !You can apply the input command to only one axis or none of them!)
                    (print !Please select your choice (only one)!))
; >>>>> continue on the right >>>>>
))
               (param_conclusions (initial_value (frame_full
                                                  (rule
                                                   (value_input_rb_rule3)))
                                                 (tcontrol_1
                                                  (rule
                                                   (value_input_rb_rule21)))
                                                 (tcontrol_2
                                                  (rule
                                                   (value_input_rb_rule22)))
                                                 (tcontrol_3
                                                  (rule
                                                   (value_input_rb_rule23)))
                                                 (ocontrol_2
                                                  (rule
                                                   (value_input_rb_rule26)))
                                                 (ocontrol_3
                                                  (rule
                                                   (value_input_rb_rule27)))
                                                 (error
                                                  (rule
                                                   (value_input_rb_rule29)))
                                                 (quaternion_initialized
                                                  (rule
                                                   (value_input_rb_rule20)))
                                                 (init_theta=0_wanted
                                                  (rule
                                                   (value_input_rb_rule20)))
                                                 (Y_matrix_full
                                                  (rule
                                                   (value_input_rb_rule24)))
                                                 (first_step_found
                                                  (rule
                                                   (value_input_rb_rule2)))
                                                 (ocontrol_1
                                                  (rule
                                                   (value_input_rb_rule25)))
                                                 (second_step_found
                                                  (rule
                                                   (value_input_rb_rule4))))
                                  (Kd_matrix (matrix_type
                                              (rule (value_input_rb_rule5)
                                                    (value_input_rb_rule8)
                                                    (value_input_rb_rule17)
                                                    (value_input_rb_rule15)))
                                             (equal_diagonal_terms
                                              (rule (value_input_rb_rule10)
                                                    (value_input_rb_rule12)))
                                             (matrix_full
```

```
                (rule (value_input_rb_rule5)
                      (value_input_rb_rule19))))
(Kp_matrix (matrix_type
                (rule (value_input_rb_rule6)
                      (value_input_rb_rule7)
                      (value_input_rb_rule14)
                      (value_input_rb_rule16)))
           (equal_diagonal_terms
                (rule (value_input_rb_rule9)
                      (value_input_rb_rule11)))
           (matrix_full
                (rule (value_input_rb_rule6)
                      (value_input_rb_rule18))))
(Kp ((1 2)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule7)))
    ((1 3)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule7)))
    ((2 1)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule7)
           (value_input_rb_rule14)))
    ((2 2)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule9)))
    ((2 3)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule7)))
    ((3 1)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule7)
           (value_input_rb_rule14)))
    ((3 2)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule7)
           (value_input_rb_rule14)))
    ((3 3)
     (rule (value_input_rb_rule6)
           (value_input_rb_rule9)))
    ((1 1) (rule (value_input_rb_rule6))))
(Kd ((1 2)
     (rule (value_input_rb_rule5)
           (value_input_rb_rule8)))
    ((1 3)
     (rule (value_input_rb_rule5)
           (value_input_rb_rule8)))
    ((2 1)
     (rule (value_input_rb_rule5)
           (value_input_rb_rule8)
           (value_input_rb_rule15)))
    ((2 2)
     (rule (value_input_rb_rule5)
           (value_input_rb_rule10)))
    ((2 3)
     (rule (value_input_rb_rule5)
           (value_input_rb_rule8)))
    ((3 1)
     (rule (value_input_rb_rule5)
           (value_input_rb_rule8)
           (value_input_rb_rule15)))
```

```
                                ((3 2)
                                 (rule (value_input_rb_rule5)
                                       (value_input_rb_rule8)
                                       (value_input_rb_rule15)))
                                ((3 3)
                                 (rule (value_input_rb_rule5)
                                       (value_input_rb_rule10)))
                                ((i 1) (rule (value_input_rb_rule5))))
                           (Y_matrix (theta_y
                                      (rule (value_input_rb_rule21)
                                            (value_input_rb_rule22)))
                                     (theta_x
                                      (rule (value_input_rb_rule21)
                                            (value_input_rb_rule22)))
                                     (omega_y
                                      (rule (value_input_rb_rule26)
                                            (value_input_rb_rule25)))
                                     (omega_x
                                      (rule (value_input_rb_rule26)
                                            (value_input_rb_rule25)))
                                     (theta_x
                                      (rule (value_input_rb_rule21)))
                                     (omega_x
                                      (rule (value_input_rb_rule25))))
                           (inertial_matrix ((3 3)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((2 2)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((1 2)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((1 3)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((2 1)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((2 3)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((3 1)
                                             (rule
                                              (value_input_rb_rule13)))
                                            ((3 2)
                                             (rule
                                              (value_input_rb_rule13))))
                                     (inertial_mat
                                      (matrix_full
                                       (rule (value_input_rb_rule13))))))
(value_input_rb_rule3 ($type (ifany))
                      ($then (conclude (initial_value frame_full) true))
                      (created_by (samir))
                      (created_on (|2_13_85|))
                      ($if ($and (triple (initial_value first_step_found)
                                         =
                                         true)
                                 (triple (response_chosen response_type)
                                         =
                                         |Step response|)
```

```
                        ($not (triple (initial_value deltaT) = nil))
                        ($not
                         (triple (initial_value steady_state_error)
                                 =
                                 nil)))
                   ($and (triple (response_chosen response_type)
                                 =
                                 |Frequency response|)
                         (triple (initial_value first_step_found)
                                 =
                                 true)
                         (triple (initial_value second_step_found)
                                 =
                                 true))
                   ($and (triple (response_chosen response_type)
                                 =
                                 |Impulse response|)
                         (triple (initial_value first_step_found)
                                 =
                                 true)
                         ($not (triple (initial_value deltaT) = nil))
                         (triple (initial_value imp_amp_found)
                                 =
                                 true)))
              (modified_by (|Juan J. Rodriguez-Moscoso|))
              (modified_on (|6_9_85|)))
(value_input_rb_rule5 ($type (ifall))
              (created_by (samir))
              (created_on (|3_12_85|))
              ($if (triple (controller_type_desired derivative) = no))
              ($then (conclude (Kd_matrix matrix_full) true)
                     (conclude (Kd_matrix matrix_type) zero)
                     (conclude (Kd (1 1)) 0.0)
                     (conclude (Kd (1 2)) 0.0)
                     (conclude (Kd (1 3)) 0.0)
                     (conclude (Kd (2 1)) 0.0)
                     (conclude (Kd (2 2)) 0.0)
                     (conclude (Kd (2 3)) 0.0)
                     (conclude (Kd (3 1)) 0.0)
                     (conclude (Kd (3 2)) 0.0)
                     (conclude (Kd (3 3)) 0.0)
                     (clause_print
                      |All values of the Kd Matrix have been found.|))
              (modified_by (|Juan J. Rodriguez-Moscoso|))
              (modified_on (|6_9_85|)))
(value_input_rb_rule6 ($type (ifall))
              (created_by (samir))
              (created_on (|3_12_85|))
              ($if
               (triple (controller_type_desired proportional) = no))
              ($then (conclude (Kp_matrix matrix_full) true)
                     (conclude (Kp_matrix matrix_type) zero)
                     (conclude (Kp (1 1)) 0.0)
                     (conclude (Kp (1 2)) 0.0)
                     (conclude (Kp (1 3)) 0.0)
                     (conclude (Kp (2 1)) 0.0)
                     (conclude (Kp (2 2)) 0.0)
                     (conclude (Kp (2 3)) 0.0)
                     (conclude (Kp (3 1)) 0.0)
                     (conclude (Kp (3 2)) 0.0)
                     (conclude (Kp (3 3)) 0.0)
```

```
                              (clause_print
                               !All values of the Kp Matrix have been found.!))
                        (modified_by (!Juan J. Rodriguez-Moscoso!))
                        (modified_on (!6_9_85!)))
(value_input_rb_rule18 ($type (ifany))
                        ($then (conclude (Kp_matrix matrix_full) true))
                        (created_by (samir))
                        (created_on (!3_15_85!))
                        ($if ($and (triple (Kp_matrix matrix_type) = diagonal)
                                   (triple (Kp_matrix equal_diagonal_terms)
                                           =
                                           true))
                             ($and (triple (Kp_matrix matrix_type) = diagonal)
                                   (triple (Kp_matrix equal_diagonal_terms)
                                           =
                                           false))
                             (triple (Kp_matrix matrix_type) = symmetric)
                             (triple (Kp_matrix matrix_type) = regular))
                        (modified_by
                         (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                        (modified_on (!6_9_85!)))
(value_input_rb_rule19 ($type (ifany))
                        ($then (conclude (Kd_matrix matrix_full) true))
                        (created_by (samir))
                        (created_on (!3_15_85!))
                        ($if ($and (triple (Kd_matrix matrix_type) = diagonal)
                                   (triple (Kd_matrix equal_diagonal_terms)
                                           =
                                           true))
                             ($and (triple (Kd_matrix matrix_type) = diagonal)
                                   (triple (Kd_matrix equal_diagonal_terms)
                                           =
                                           false))
                             (triple (Kd_matrix matrix_type) = symmetric)
                             (triple (Kd_matrix matrix_type) = regular))
                        (modified_by
                         (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                        (modified_on (!6_9_85!)))
(value_input_rb_rule21 ($type (ifall))
                        (created_by (samir))
                        (created_on (!4_30_85!))
                        ($if (triple (initial_value init_theta=0_wanted) = yes))
                        ($then (conclude (initial_value tcontrol_1) true)
                               (conclude (Y_matrix theta_x) 0.0)
                               (conclude (Y_matrix theta_y) 0.0)
                               (conclude (Y_matrix theta_z) 0.0)
                               (clause_print
                                !The angular position for all axes have been initialized.!))
                        (modified_by
                         (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                        (modified_on (!6_9_85!)))
(value_input_rb_rule22 ($type (ifall))
                        (created_by (samir))
                        (created_on (!4_30_85!))
                        ($if (triple (initial_value
                                      equal_theta_values_for_all_axes)
                                     =
                                     yes)
                             (clause_print
                              !The common value of angular position for all axes, Theta(x), is!)
                             (clause_print required.))
```

```
                        ($not (triple (Y_matrix theta_z) = nil)))
                    ($then (conclude (initial_value tcontrol_2) true)
                           (conclude (Y_matrix theta_y) (Y_matrix theta_x))
                           (conclude (Y_matrix theta_z) (Y_matrix theta_x))
                           (clause_print
                            !Angular position has been initialized.!))
                    (modified_by
                     (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (modified_on (!6_9_85!)))
(value_input_rb_rule23 ($type (ifall))
                    (created_by (samir))
                    (created_on (!4_30_85!))
                    ($if (triple (initial_value
                                  equal_theta_values_for_all_axes)
                                 =
                                 no)
                         (clause_print
                          !Values for angular position (Theta) along all axes are required.!)
                         ($not (triple (Y_matrix theta_x) = nil))
                         ($not (triple (Y_matrix theta_y) = nil))
                         ($not (triple (Y_matrix theta_z) = nil)))
                    ($then (conclude (initial_value tcontrol_3) true)
                           (clause_print
                            !Angular position has been initialized.!))
                    (modified_by
                     (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (modified_on (!6_9_85!)))
(value_input_rb_rule26 ($type (ifall))
                    (created_by (samir))
                    (created_on (!4_30_85!))
                    ($if (triple (initial_value
                                  equal_omega_values_for_all_axes)
                                 =
                                 yes)
                         (clause_print
                          !The common value of angular rate, Omega(x), is required.!)
                         ($not (triple (Y_matrix omega_x) = nil)))
                    ($then (conclude (initial_value ocontrol_2) true)
                           (conclude (Y_matrix omega_y) (Y_matrix omega_x))
                           (conclude (Y_matrix omega_z) (Y_matrix omega_x))
                           (clause_print
                            !The angular rate has been initialized.!))
                    (modified_by
                     (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (modified_on (!6_9_85!)))
(value_input_rb_rule27 ($type (ifall))
                    (created_by (samir))
                    (created_on (!4_30_85!))
                    ($if (triple (initial_value
                                  equal_omega_values_for_all_axes)
                                 =
                                 no)
                         (clause_print
                          !Values for angular rate along all axes are required.!)
                         ($not (triple (Y_matrix omega_x) = nil))
                         ($not (triple (Y_matrix omega_y) = nil))
                         ($not (triple (Y_matrix omega_z) = nil)))
                    ($then (conclude (initial_value ocontrol_3) true)
                           (clause_print
                            !The angular rate has been initialized.!))
                    (modified_by
```

```
                        (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                       (modified_on (!6_9_85!)))
(value_input_rb_rule29 ($type (ifall))
                       ($if
                        (triple (initial_value integration_method) = Euler))
                       ($then (conclude (initial_value error) 0.0))
                       (created_by (!Bor-Jau Hsieh & Juan Rodriguez-Moscoso!))
                       (created_on (!6_12_85!)))
(value_input_rb_rule20 ($type (ifall))
                       ($if (triple (quaternion wanted_by_user) = yes)
                            ($not (triple (quaternion roll_angle) = nil))
                            ($not (triple (quaternion pitch_angle) = nil))
                            ($not (triple (quaternion yaw_angle) = nil)))
                       ($else
                        (conclude (initial_value quaternion_initialized) true))
                       (created_by (samir))
                       (created_on (!4_29_85!))
                       ($then (conclude (initial_value quaternion_initialized)
                                        true)
                              (conclude (initial_value init_theta=0_wanted)
                                        yes)
                              (clause_print
                               !The Roll, Pitch, and Yaw angles have been initialized.!))
                       (modified_by (!Bor-Jau Hsieh & Juan Rodriguez-Moscoso!))
                       (modified_on (!6_12_85!)))
(value_input_rb_rule24 ($then (conclude (initial_value Y_matrix_full) true))
                       (created_by (samir))
                       (created_on (!4_30_85!))
                       ($type (ifall))
                       ($if (clause_print
                             !Initial values for angular rate and/or angular position are required!)
                            ($or (triple (initial_value tcontrol_1) = true)
                                 (triple (initial_value tcontrol_2) = true)
                                 (triple (initial_value tcontrol_3) = true))
                            ($or (triple (initial_value ocontrol_1) = true)
                                 (triple (initial_value ocontrol_2) = true)
                                 (triple (initial_value ocontrol_3) = true)))
                       (modified_by (likes_Tfinal_value))
                       (modified_on (!6_14_85!)))
(value_input_rb_rule8 ($type (ifall))
                      (created_by (samir))
                      (created_on (!3_12_85!))
                      ($then (conclude (Kd_matrix matrix_type) diagonal)
                             (conclude (Kd (1 2)) 0.0)
                             (conclude (Kd (1 3)) 0.0)
                             (conclude (Kd (2 1)) 0.0)
                             (conclude (Kd (2 3)) 0.0)
                             (conclude (Kd (3 1)) 0.0)
                             (conclude (Kd (3 2)) 0.0))
                      ($if
                       (triple (controller_type_desired
                                diff_cross_coup_bet_axes)
                               =
                               no))
                      (modified_by (likes_Tfinal_value))
                      (modified_on (!6_14_85!)))
(value_input_rb_rule10 ($type (ifall))
                       (created_by (samir))
                       (created_on (!3_12_85!))
                       ($then (conclude (Kd_matrix equal_diagonal_terms) true)
                              (conclude (Kd (2 2)) (Kd (1 1)))
```

```
                        (conclude (Kd (3 3)) (Kd (1 1)))
                        (clause_print
                         !All values of the Kd Matrix have been found.!))
                    ($if (triple (controller_type_desired
                                  diff_cross_coup_bet_axes)
                                 =
                                 no)
                         (triple (controller_type_desired
                                  same_differential_control_for_all_axes)
                                 =
                                 yes)
                         (clause_print
                          !The value of the Kd Controller Gain Matrix common to all axes!)
                         (clause_print !is required.!)
                         ($not (triple (Kd (1 1)) = nil)))
                    (modified_by (likes_Tfinal_value))
                    (modified_on (!6_14_85!)))
(value_input_rb_rule17 ($type (ifall))
                    (created_by (samir))
                    (created_on (!3_14_85!))
                    ($then (conclude (Kd_matrix matrix_type) regular)
                           (clause_print
                            !All values of the Kd Matrix have been found.!))
                    ($if (triple (controller_type_desired
                                  diff_cross_coup_bet_axes)
                                 =
                                 yes)
                         (triple (Kd_matrix matrix_symmetric) = no)
                         (clause_print
                          !Since Kd is neither symmetric nor diagonal, all 9 of its values!)
                         (clause_print
                          !are required. As you see them prompted for, please enter values.!)
                         ($not (triple (Kd (1 1)) = nil))
                         ($not (triple (Kd (1 2)) = nil))
                         ($not (triple (Kd (1 3)) = nil))
                         ($not (triple (Kd (2 1)) = nil))
                         ($not (triple (Kd (2 2)) = nil))
                         ($not (triple (Kd (2 3)) = nil))
                         ($not (triple (Kd (3 1)) = nil))
                         ($not (triple (Kd (3 2)) = nil))
                         ($not (triple (Kd (3 3)) = nil)))
                    (modified_by (likes_Tfinal_value))
                    (modified_on (!6_14_85!)))
(value_input_rb_rule12 ($type (ifall))
                    (created_by (samir))
                    (created_on (!3_12_85!))
                    ($then (conclude (Kd_matrix equal_diagonal_terms) false)
                           (clause_print
                            !All values of the Kd Controller Gain Matrix have been found.!))
                    ($if (triple (controller_type_desired
                                  diff_cross_coup_bet_axes)
                                 =
                                 no)
                         (triple (controller_type_desired
                                  same_differential_control_for_all_axes)
                                 =
                                 no)
                         (clause_print
                          !The different values of the Kd Controller Gain Matrix along!)
                         (clause_print
                          !the axes are required. Gains along the X, Y, and Z axes will!)
```

```
                        (clause_print
                         !be prompted in that order. Please enter values.!)
                        ($not (triple (Kd (1 1)) = nil))
                        ($not (triple (Kd (2 2)) = nil))
                        ($not (triple (Kd (3 3)) = nil)))
                       (modified_by (likes_Tfinal_value))
                       (modified_on (!6_14_85!)))
(value_input_rb_rule2 ($type (ifall))
                      (created_by (samir))
                      (created_on (!2_5_85!))
                      ($then (conclude (initial_value first_step_found) true))
                      ($if (triple (inertial_mat matrix_full) = true)
                           (triple (Kp_matrix matrix_full) = true)
                           (triple (Kd_matrix matrix_full) = true)
                           (triple (initial_value quaternion_initialized)
                                   =
                                   true)
                           ($not
                            (triple (initial_value integration_method) = nil))
                           ($not
                            (triple (response_chosen response_type) = nil))
                           ($not
                            (triple (initial_value axis_of_input_command)
                                    =
                                    nil))
                           ($not (triple (initial_value TO) = nil))
                           ($not (triple (initial_value error) = nil))
                           (triple (initial_value Y_matrix_full) = true))
                      (modified_by (!Samir, Andy, and Juan!))
                      (modified_on (!6_15_85!)))
(value_input_rb_rule25 ($if (triple (initial_value init_omega=0_wanted) = yes))
                       (created_by (samir))
                       (created_on (!4_30_85!))
                       ($type (ifall))
                       ($then (conclude (initial_value ocontrol_1) true)
                              (conclude (Y_matrix omega_x) 0.0)
                              (conclude (Y_matrix omega_y) 0.0)
                              (conclude (Y_matrix omega_z) 0.0)
                              (clause_print
                               !The angular rate has been initialized.!))
                       (modified_by
                        (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                       (modified_on (!6_16_85!)))
(value_input_rb_rule1 ($type (ifall))
                      ($if (triple (initial_value frame_full) = true))
                      (created_by (samir))
                      (created_on (!2_5_85!))
                      ($then (clause_print
                              !All parameter values needed for the simulation program have been found.!)
                             (clause_print
                              !By using option 3 of the  TOP LEVEL MENU, the simulation program can be!)
                             (clause_print run.))
                      (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                      (modified_on (!6_16_85!)))
(value_input_rb_rule7 ($type (ifall))
                      (created_by (samir))
                      (created_on (!3_12_85!))
                      ($then (conclude (Kp_matrix matrix_type) diagonal)
                             (conclude (Kp (1 2)) 0.0)
                             (conclude (Kp (1 3)) 0.0)
                             (conclude (Kp (2 1)) 0.0)
```

```
                        (conclude (Kp (2 3)) 0.0)
                        (conclude (Kp (3 1)) 0.0)
                        (conclude (Kp (3 2)) 0.0))
                    ($if
                     (triple (controller_type_desired
                              prop_cross_coup_bet_axes)
                             =
                             no))
                    (modified_by (|Juan J. Rodriguez-Moscoso|))
                    (modified_on (|6_16_85|)))
(value_input_rb_rule9 ($type (ifall))
                    (created_by (samir))
                    (created_on (|3_12_85|))
                    ($then (conclude (Kp_matrix equal_diagonal_terms) true)
                           (conclude (Kp (2 2)) (Kp (1 1)))
                           (conclude (Kp (3 3)) (Kp (1 1)))
                           (clause_print
                            |All values of the Kp Matrix have been found.|))
                    ($if (triple (controller_type_desired
                                  prop_cross_coup_bet_axes)
                                 =
                                 no)
                         (triple (controller_type_desired
                                  same_proportional_control_for_all_axes)
                                 =
                                 yes)
                         (clause_print
                          |The common value of the Kp gain for all axes is required.  It is|)
                         (clause_print |called Kp (1 1).|)
                         ($not (triple (Kp (1 1)) = nil)))
                    (modified_by (|Juan J. Rodriguez-Moscoso|))
                    (modified_on (|6_16_85|)))
(value_input_rb_rule11 ($type (ifall))
                     (created_by (samir))
                     (created_on (|3_12_85|))
                     ($then (conclude (Kp_matrix equal_diagonal_terms) false)
                            (clause_print
                             |All values of the Kp Matrix have been found.|))
                     ($if (triple (controller_type_desired
                                   prop_cross_coup_bet_axes)
                                  =
                                  no)
                          (triple (controller_type_desired
                                   same_proportional_control_for_all_axes)
                                  =
                                  no)
                          (clause_print
                           |The values of the elements of the Kp Matrix along the axes are|)
                          (clause_print required.)
                          ($not (triple (Kp (1 1)) = nil))
                          ($not (triple (Kp (2 2)) = nil))
                          ($not (triple (Kp (3 3)) = nil)))
                     (modified_by (|Juan J. Rodriguez-Moscoso|))
                     (modified_on (|6_16_85|)))
(value_input_rb_rule14 ($type (ifall))
                     (created_by (samir))
                     (created_on (|3_14_85|))
                     ($then (conclude (Kp_matrix matrix_type) symmetric)
                            (conclude (Kp (2 1)) (Kp (1 2)))
                            (conclude (Kp (3 1)) (Kp (1 3)))
                            (conclude (Kp (3 2)) (Kp (2 3)))
```

```
                        (clause_print
                         !All values of the Kp Matrix have been found.!))
                    ($if (triple (controller_type_desired
                                  prop_cross_coup_bet_axes)
                                 =
                                 yes)
                         (triple (Kp_matrix matrix_symmetric) = yes)
                         (clause_print
                          !Since Kp is a symmetric matrix, only 6 of its values are required.!)
                         (clause_print
                          !As you see them prompted for, please enter values.!)
                         ($not (triple (Kp (1 1)) = nil))
                         ($not (triple (Kp (1 2)) = nil))
                         ($not (triple (Kp (1 3)) = nil))
                         ($not (triple (Kp (2 2)) = nil))
                         ($not (triple (Kp (2 3)) = nil))
                         ($not (triple (Kp (3 3)) = nil)))
                    (modified_by (!Juan J. Rodriguez-Moscoso!))
                    (modified_on (!6_16_85!)))
(value_input_rb_rule16 ($type (ifall))
                    (created_by (samir))
                    (created_on (!3_14_85!))
                    ($then (conclude (Kp_matrix matrix_type) regular)
                           (clause_print
                            !All values of the Kp Matrix have been found.!))
                    ($if (triple (controller_type_desired
                                  prop_cross_coup_bet_axes)
                                 =
                                 yes)
                         (triple (Kp_matrix matrix_symmetric) = no)
                         (clause_print
                          !Since Kp is neither symmetric nor diagonal, all 9 of its values!)
                         (clause_print
                          !are required. As you see them prompted for, please enter values.!)
                         ($not (triple (Kp (1 1)) = nil))
                         ($not (triple (Kp (1 2)) = nil))
                         ($not (triple (Kp (1 3)) = nil))
                         ($not (triple (Kp (2 1)) = nil))
                         ($not (triple (Kp (2 2)) = nil))
                         ($not (triple (Kp (2 3)) = nil))
                         ($not (triple (Kp (3 1)) = nil))
                         ($not (triple (Kp (3 2)) = nil))
                         ($not (triple (Kp (3 3)) = nil)))
                    (modified_by (!Juan J. Rodriguez-Moscoso!))
                    (modified_on (!6_16_85!)))
(value_input_rb_rule4 ($type (ifall))
                   (created_by (samir))
                   (created_on (!2_13_85!))
                   ($then (conclude (initial_value second_step_found) true)
                          (clause_print
                           !All parameter values for FREQUENCY response analysis have been!)
                          (clause_print found.))
                   ($if (clause_print
                         !Initial parameter values for FREQUENCY response analysis are required.!)
                        ($not (triple (initial_value amplitude) = nil))
                        ($not
                         (triple (initial_value init_frequency_value)
                                 =
                                 nil))
                        ($not
                         (triple (initial_value
```

```
                                frequency_deltaT)
                               =
                               nil))
                        ($not
                         (triple (initial_value number_of_decades) = nil))
                        ($not
                         (triple (initial_value
                                  number_of_sampling_frequency_per_decade)
                                 =
                                 nil))
                        ($not (triple (initial_value phase) = nil)))
                    (modified_by (|Juan J. Rodriguez-Moscoso|))
                    (modified_on (|6_16_85|)))
(value_input_rb_rule13 ($type (ifall))
                       (created_by (samir))
                       (created_on (|3_13_85|))
                       ($if (clause_print
                             |For the present,  the Inertia matrix is assumed to be diagonal.|)
                            (clause_print
                             |It is also assumed  all diagonal  elements to be 'equal'.  This|)
                            (clause_print
                             |assumption makes the 'system' uncoupled in nature.|)
                            (clause_print
                             |Please enter the common element of the diagonal of the Inertia|)
                            (clause_print Matrix.)
                            ($not (triple (inertial_matrix (1 1)) = nil)))
                       ($then (conclude (inertial_matrix (3 3))
                                        (inertial_matrix (1 1)))
                              (conclude (inertial_matrix (2 2))
                                        (inertial_matrix (1 1)))
                              (conclude (inertial_mat matrix_full) true)
                              (conclude (inertial_matrix (1 2)) 0.0)
                              (conclude (inertial_matrix (1 3)) 0.0)
                              (conclude (inertial_matrix (2 1)) 0.0)
                              (conclude (inertial_matrix (2 3)) 0.0)
                              (conclude (inertial_matrix (3 1)) 0.0)
                              (conclude (inertial_matrix (3 2)) 0.0)
                              (clause_print
                               |All values of the 'Inertia matrix' have been found.|))
                       (modified_by (|Juan J. Rodriguez-Moscoso|))
                       (modified_on (|6_16_85|)))
(value_input_rb_rule15 ($type (ifall))
                       (created_by (samir))
                       (created_on (|3_14_85|))
                       ($then (conclude (Kd_matrix matrix_type) symmetric)
                              (conclude (Kd (2 1)) (Kd (1 2)))
                              (conclude (Kd (3 1)) (Kd (1 3)))
                              (conclude (Kd (3 2)) (Kd (2 3)))
                              (clause_print
                               |All values of the Kd Matrix have been found.|))
                       ($if (triple (controller_type_desired
                                     diff_cross_coup_bet_axes)
                                    =
                                    yes)
                            (triple (Kd_matrix matrix_symmetric) = yes)
                            (clause_print
                             |Since Kd is a symmetric matrix, only 6 of its values are required.|)
                            (clause_print
                             |As you see them prompted for, please enter values.|)
                            ($not (triple (Kd (1 1)) = nil))
                            ($not (triple (Kd (1 2)) = nil))
```

```
        ($not (triple (Kd (1 3)) = nil))
        ($not (triple (Kd (2 2)) = nil))
        ($not (triple (Kd (2 3)) = nil))
        ($not (triple (Kd (3 3)) = nil)))
    (modified_by (|Juan J. Rodriguez-Moscoso|))
    (modified_on (16_16_851)))
```

```
(file_index (frames.(file_index (simxpert.1))
                    (param_specs (simxpert.1))
                    (system_specs (simxpert.1))
                    (value_input_rb (rules.1))
                    (sim_expert_agenda (simxpert.1))
                    (start_agenda (simxpert.1))
                    (top_level_agenda (simxpert.1))
                    (top_level_menu (simxpert.1))
                    (top_level_control (simxpert.1))
                    (stage_1_rb (simxpert.1))
                    (change_param_control (simxpert.1))
                    (disp_init_val_control (simxpert.1))
                    (param_menu (simxpert.1))
                    (disp_init_val_agenda (simxpert.1))
                    (disp_init_val_rb (simxpert.1))
                    (change_param_agenda (simxpert.1))
                    (test_rb (as.1))
                    (output_display_menu (simxpert.1))
                    (output_display_control (simxpert.1))
                    (output_display_agenda (simxpert.1))
                    (output_display_rb (simxpert.1))
                    (run_rb (simxpert.1))
                    (run_agenda (simxpert.1))
                    (user_rb (user.1))
                    (change_param_rb (change.1))
                    (change_matrix_rb (change.1)))
            (files (simxpert.1 (file_index)
                               (system_specs)
                               (param_specs)
                               (sim_expert_agenda)
                               (start_agenda)
                               (top_level_agenda)
                               (top_level_menu)
                               (top_level_control)
                               (stage_1_rb (rule_base))
                               (change_param_control)
                               (disp_init_val_control)
                               (param_menu)
                               (disp_init_val_agenda)
                               (disp_init_val_rb (rule_base))
                               (change_param_agenda)
                               (output_display_menu)
                               (output_display_control)
                               (output_display_agenda)
                               (output_display_rb (rule_base))
                               (run_rb (rule_base))
                               (run_agenda))
                   (rules.1 (value_input_rb (rule_base)))
                   (user.1 (user_rb (rule_base)))
                   (user1.1 (user_test_rb (rule_base)))
                   (change.1 (change_param_rb (rule_base))
                             (change_matrix_rb (rule_base)))))
(system_specs (weights (default ((5 5))))
              (param_default (message_format (print
                                              !Please, enter the following parameter value:!)
                                             (path))
                             (prompt_format (values))
                             (param_cf (no))
                             (value_type (numerical))
                             (value_multiplicity (single))
                             (value_required (yes))
```

```
                    (find_strategy (fget)
                                   (try_all)
                                   (ask)
                                   (default)
                                   (unknown)))
               (backup_commands (b) (back) (backup) (up) (^) (undo))
               (help_commands (help) (?) (hlep) (ehlp))
               (why_commands (why) (why?)))
(param_specs (CF (default (1.0))
                 (unknown (0.0))
                 (value_type (numerical))
                 (range ((>= 0.0)) ((<= 1.0))))
             (T0 (param_cf (no))
                 (message_format (print
                                  !The time of application of the input signal, also known as!)
                                 (print
                                  !the initial time,  is required.  Usually its  value is 0.0!)
                                 (print seconds.)
                                 (path)))
             (Tfinal
              (message_format (print
                               !The time at which the 'simulation program' should stop computing!)
                              (print
                               !the state of the system is required.  It is also known as Tfinal.!)
                              (path)))
             (proportional (value_required (yes))
                           (message_format (print
                                            !Do you want a proportional (P) type control?!)
                                           (print !(For P and PD type control)!)))
                           (prompt_format (menu_input))
                           (legal_values (yes) (no)))
             (derivative (value_required (yes))
                         (legal_values (yes) (no))
                         (message_format (print
                                          !Do you want a differential (D) type control?!)
                                         (print !(For D and PD type control)!)))
                         (prompt_format (menu_input)))
             (number_of_sampling_frequency_per_decade
; <<<<< start back on the left <<<<<
    (message_format (print
                     !Enter the number of sampling frequencies per decade!)
                    (print !for FREQUENCY response!)
                    (print !Normal value is 3 frequencies!)))
; >>>>> continue on the right >>>>>

                                                 (prompt_format (values)))
             (number_of_decades (message_format (print
                                                 !Enter the number of decades of input signal for FREQUENCY response!)
                                                (print
                                                 !Normal value is 3 decades!)))
                                (prompt_format (values)))
             (matrix_symmetric (value_required (yes))
                               (legal_values (yes) (no))
                               (message_format (print
                                                !Is the following matrix symmetric?!)
                                               (path))
                               (prompt_format (menu_input)))
             (same_proportional_control_for_all_axes (value_required (yes))
                                                     (legal_values (yes) (no))
                                                     (message_format (print
                                                                      !Do you want the same proportional controller gain along!)
```

```
                                                    (print
                                                     !all axes ?!))
                                          (prompt_format
                                           (menu_input)))
(same_differential_control_for_all_axes (value_required (yes))
                                          (legal_values (yes) (no))
                                          (message_format (print
                                                           !Do you want the same differential controller gain for!)
                                                          (print
                                                           !all axes ?!))
                                          (prompt_format
                                           (menu_input)))
(response_type (value_type (literal))
               (value_multiplicity (single))
               (value_required (yes))
               (legal_values (!Step response!) (!Frequency response!))
               (message_format (print
                                !Please select from the following menu the type of System!)
                               (print
                                !response. Please make only one choice.!))
               (prompt_format (menu_input)))
(likes_Tfinal_value (value_type (literal))
                    (value_multiplicity (single))
                    (value_required (yes))
                    (legal_values (yes) (no))
                    (message_format (print
                                     !Just above is the value of final time (Tfinal) as computed by!)
                                    (print
                                     !the system. Do you think that this value is what you want ?.!)
                                    (print
                                     !If you answer no then you can provide a new value for Tfinal.!))
                    (prompt_format (menu_input)))
(frame_full)
(wanted_by_user (value_type (literal))
                (value_multiplicity (single))
                (value_required (yes))
                (legal_values (yes) (no))
                (message_format
                 (print
                  !Do you want the Quaternion block to be included in the simulation?!))
                (prompt_format (menu_input)))
(roll_angle (prompt_format (values))
            (value_multiplicity (single))
            (message_format (print
                             !Please enter initial roll angle (rotation about the X axis)!)
                            (print !Enter your value in radians!)))
(pitch_angle (prompt_format (values))
             (message_format (print
                              !Please enter initial pitch angle (rotation about the Z axis)!)
                             (print !Enter your value in radians!)))
(yaw_angle (prompt_format (values))
           (message_format (print
                            !Please enter initial Yaw angle (rotation about Y axis)!)
                           (print !Enter your value in radians!)))
(init_theta=0_wanted (value_type (literal))
                     (value_multiplicity (single))
                     (value_required (yes))
                     (message_format (print
                                      !Do you want the initial value of angular position to be 0.0 for!)
                                     (print !all axes?!))
                     (prompt_format (menu_input))
```

```
                    (legal_values (yes) (no)))
(init_omega=0_wanted (value_type (literal))
                     (value_multiplicity (single))
                     (value_required (yes))
                     (legal_values (yes) (no))
                     (message_format (print
                                      !Do you want the initial value of the angular velocity to be 0.0!)
                                     (print !all axes?!))
                     (prompt_format (menu_input)))
(equal_theta_values_for_all_axes (value_type (literal))
                                 (value_multiplicity (single))
                                 (value_required (yes))
                                 (legal_values (yes) (no))
                                 (message_format (print
                                                  !Do you want the angular position (Theta) to have the same initial!)
                                                 (print
                                                  !value for all axes?!))
                                 (prompt_format (menu_input)))
(equal_omega_values_for_all_axes (value_type (literal))
                                 (value_multiplicity (single))
                                 (value_required (yes))
                                 (legal_values (yes) (no))
                                 (message_format (print
                                                  !Do you want the angular velocity (Omega) to have the same initial!)
                                                 (print
                                                  !value for all axes?!))
                                 (prompt_format (menu_input)))
(integration_method (value_type (literal))
                    (value_multiplicity (single))
                    (value_required (yes))
                    (legal_values (Euler)
                                  (!Fourth-order Runge-Kutta!)
                                  (!Predictor corrector!))
                    (message_format
                     (print
                      !Please, select one method of integration:!))
                    (prompt_format (menu_input)))
(deltaT (message_format (print
                         !Please, enter a value for delta_T. This is the value of the time!)
                        (print !increment in the simulation.!))
        (prompt_format (values))
        (range ((> 0.0)) ((< 0.1))))
(error (range ((> 0.0)))
       (message_format (print
                        !Enter the precision wanted for multi-step integration selected.!)
                       (print
                        !Normal values range from 0.0001 (normal precision) to 1.E-10 !)
                       (print !(double precision).!)))
(wants_to_run_simulation (value_type (literal))
                         (value_multiplicity (single))
                         (value_required (yes))
                         (legal_values (yes) (no))
                         (message_format (print)
                                         (print
                                          !Do you want to run the Simulation Program?!)
                                         (print
                                          !  - If you say NO, you can go back and review parameter values.!)
                                         (print
                                          !  - If YES, you will be running the FORTRAN-77 simulation program.!))
                         (prompt_format (menu_input)))
(steady_state_error (message_format (print
```

```
                                        !Please, enter the steady state error in percentage. For example,!)
                                       (print
                                        !type in 2.0 for a 2% steady state error.!))
                              (prompt_format (values)))
            (amplitude (message_format (print
                                        !Enter the Amplitude of the input waveform for FREQUENCY response!)
                                       (print !Normal value is 1.0 radian!))
                       (prompt_format (values)))
            (frequency_deltaT (prompt_format (menu_input))
                              (legal_values (256) (512) (1024))
                              (message_format (print
                                               !Please enter a value for deltaT for frequency response.!)
                                              (print
                                               !If you choose 512 from the menu below deltaT will be fixed!)
                                              (print
                                               !to 1/512 or 0.00195312 sec.!)
                                              (print
                                               !This value of 512 is recommended.!)))
            (phase (message_format (print
                                    !Enter the phase of the input waveform for FREQUENCY response!)
                                   (print !Normal value is 0.0 radians!))
                   (prompt_format (values)))
            (init_frequency_value (message_format (print
                                                   !Enter the lowest frequency of input wavwform to be applied to!)
                                                  (print
                                                   !the system for FREQUENCY response!)
                                                  (print
                                                   !Enter your value in Hertz!)
                                                  (print
                                                   !Normal value is 0.0159154 Hz corresponding to 0.1 rad/sec!))
                                  (prompt_format (values)))
            (diff_cross_coup_bet_axes (value_multiplicity (single))
                                      (value_required (yes))
                                      (legal_values (yes) (no))
                                      (message_format
                                       (print
                                        !For differential control do you want cross-coupling between control axes?!))
                                      (prompt_format (menu_input)))
            (prop_cross_coup_bet_axes (value_multiplicity (single))
                                      (value_required (yes))
                                      (legal_values (yes) (no))
                                      (message_format
                                       (print
                                        !For proportional control, do you want cross-coupling between control axes?!))
                                      (prompt_format (menu_input)))
            (param_specs))
(sim_expert_agenda ((frem 'Y_matrix))
                   ((frem 'quaternion))
                   ((frem 'controller_type_desired))
                   ((frem 'inertial_mat))
                   ((frem 'Kd_matrix))
                   ((frem 'Kp_matrix))
                   ((frem 'inertial_matrix))
                   ((frem 'Kd))
                   ((frem 'Kp))
```

```
                    ((frem 'initial_value))
                    ((reset_rule_base 'value_input_rb))
                    ((back '(value_input_rb_rule1)
                           'value_input_rb)))
(start_agenda ((array eigi flonum-block 6))
              ((array eigr flonum-block 6))
              ((array one_over_tau flonum-block 1))
              ((set_up_frames_from_file 'rules.1))
              ((set_up_frames_from_file 'change.1))
              ((include demonar.1))
              ((array inertial_matrix flonum-block 3 3))
              ((array Kp flonum-block 3 3))
              ((array Kd flonum-block 3 3))
              ((cfasl 'eigenval.o
                      '_eigen_
                      'eigen
                      "real-function"
                      "-lF77"))
              ((include demon03.1))
              ((include demon02.1))
              ((include demon01.1))
              ((include demon00.1))
              ((terpri))
              ((terpri))
              ((msg N
                    "This Expert System provides an intelligent interface  to a generic"))
              ((msg N
                    "simulation program for spacecraft attitude control problems. Below"))
              ((msg N
                    "is a menu of  the functions  the system can perform.  Control will"))
              ((msg N
                    "repeatedly return to this menu after executing each user request."))
              ((setq zzx (wait_for_user)))
              ((start_&_iterate)))
(top_level_agenda ((reset_rule_base 'stage_1_rb))
                  ((frem 'top_level_choice))
                  ((menu_input 'top_level_control))
                  ((forward 'stage_1_rb)))
(top_level_menu (!Exit to GENIE!)
                (!Set up initial parameter values!)
                (!Run simulation program!)
                (!Display current parameter values required for simulation!)
                (!Display outputs generated by simulation!)
                (!Change initial parameter values required for simulation!)
                (!Set up initial parameter values to default values!)
                (!Store current parameter values in a disk file!))
(top_level_control ((clear_display))
                   (print)
                   (print !**  Please make only one choice at a time  **!)
                   (prompt_specs (get_alternatives_from (top_level_menu))
                                 (put_data_in (top_level_choice))
                                 (data_input)))
(stage_1_rb (if_removed (d_remove_rules))
            (created_by (samir))
            (created_on (!2_20_85!))
            (rules (stage_1_rb_rule1)
                   (stage_1_rb_rule2)
                   (stage_1_rb_rule5)
                   (stage_1_rb_rule8)
                   (stage_1_rb_rule12)
                   (stage_1_rb_rule7)
```

```
                    (stage_1_rb_rule10)
                    (stage_1_rb_rule9)
                    (stage_1_rb_rule3)
                    (stage_1_rb_rule13)
                    (stage_1_rb_rule11)
                    (stage_1_rb_rule6)
                    (stage_1_rb_rule4))
               (params_in_ifs (top_level_choice
                               (top_level_menu
                                (rule (stage_1_rb_rule1)
                                      (stage_1_rb_rule2)
                                      (stage_1_rb_rule5)
                                      (stage_1_rb_rule8)
                                      (stage_1_rb_rule12)
                                      (stage_1_rb_rule7)
                                      (stage_1_rb_rule10)
                                      (stage_1_rb_rule9)
                                      (stage_1_rb_rule3)
                                      (stage_1_rb_rule13)
                                      (stage_1_rb_rule11)
                                      (stage_1_rb_rule6)
                                      (stage_1_rb_rule4))))
                              (user
                               (simulation_run
                                (rule (stage_1_rb_rule1) (stage_1_rb_rule9))))
                              (initial_value
                               (frame_full
                                (rule (stage_1_rb_rule2)
                                      (stage_1_rb_rule5)
                                      (stage_1_rb_rule8)
                                      (stage_1_rb_rule12)
                                      (stage_1_rb_rule7)
                                      (stage_1_rb_rule10)
                                      (stage_1_rb_rule3)
                                      (stage_1_rb_rule13)
                                      (stage_1_rb_rule11)
                                      (stage_1_rb_rule4))))))
(stage_1_rb_rule1 ($type (ifall))
                  (created_by (samir))
                  (created_on (|4_4_85|))
                  ($if (triple (top_level_choice top_level_menu)
                               =
                               |Display outputs generated by simulation|)
                       (triple (user simulation_run) = error_free))
                  ($then (frem 'output_display_choice)
                         (loop 'out_disp))
                  (modified_by (juan))
                  (modified_on (|6_7_85|)))
(stage_1_rb_rule2 ($type (ifall))
                  (created_by (samir))
                  (created_on (|2_20_85|))
                  ($then (eval_agenda 'run_agenda))
                  ($if (triple (top_level_choice top_level_menu)
                               =
                               |Run simulation program|)
                       (triple (initial_value frame_full) = true))
                  (modified_by (juan))
                  (modified_on (|6_7_85|)))
(stage_1_rb_rule5 ($type (ifall))
                  ($if (triple (top_level_choice top_level_menu)
                               =
```

```
                        !Run simulation program!)
                    ($not (triple (initial_value frame_full) = true)))
                   (created_by (samir))
                   (created_on (!4_29_85!))
                   ($then (clause_print
                           !The simulation program cannot be run unless the initial parameter!)
                          (clause_print
                           !values needed  for the simulation are  given.  You can use either!)
                          (clause_print !option 2 or 7 of the TOP LEVEL MENU. !))
                   (modified_by (!Juan J. Rodriguez-Moscoso!))
                   (modified_on (!6_11_85!)))
(stage_1_rb_rule8 ($type (ifall))
                   (created_by (!Juan J. Rodriguez-Moscoso!))
                   (created_on (!6_12_85!))
                   ($if (triple (top_level_choice top_level_menu)
                                =
                                !Set up initial parameter values!)
                        (triple (initial_value frame_full) = true))
                   ($then (reset_rule_base 'user_rb)
                          (back '(user_rb_rule1) 'user_rb))
                   (modified_by (!Juan J. Rodriguez-Moscoso!))
                   (modified_on (!6_12_85!)))
(stage_1_rb_rule12 ($type (ifall))
                    (created_by (!Juan J. Rodriguez-Moscoso!))
                    (created_on (!6_11_85!))
                    ($if (triple (top_level_choice top_level_menu)
                                 =
                                 !Set up initial parameter values!)
                         ($not (triple (initial_value frame_full) = true)))
                    ($then (eval_agenda 'sim_expert_agenda)
                           (frem 'top_level_choice))
                    (modified_by (!Juan J. Rodriguez-Moscoso!))
                    (modified_on (!6_12_85!)))
(stage_1_rb_rule7 ($type (ifall))
                   (created_by (samir))
                   (created_on (!2_25_85!))
                   ($if (triple (top_level_choice top_level_menu)
                                =
                                !Change initial parameter values required for simulation!)
                        (triple (initial_value frame_full) = true))
                   ($then (clause_print
                           !You will see a menu of the parameters that can be changed. By using!)
                          (clause_print
                           !option 1 of this menu you can return to the TOP LEVEL MENU. !)
                          (clause_print)
                          (clause_print !** BE CAREFULL **!)
                          (clause_print
                           ! - The values you enter will be passed on to the simulation program!)
                          (clause_print !   if you run it immediately. !)
                          (clause_print
                           ! - There is no  intelligence  present in this function.  So, if you!)
                          (clause_print
                           !   type in  a wrong value an error might occur during the execution!)
                          (clause_print !   of the simulation program. !)
                          (frem 'change_param_choice)
                          (loop 'change))
                   (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                   (modified_on (!6_13_85!)))
(stage_1_rb_rule10 ($type (ifall))
                    (created_by (samir))
                    (created_on (!5_5_85!))
```

```
                    ($if (triple (top_level_choice top_level_menu)
                                 =
                                 !Change initial parameter values required for simulation!)
                         ($not (triple (initial_value frame_full) = true)))
                    ($then (clause_print
                            !There are no  values in the database.  Please use  option 2 or 7 of!)
                           (clause_print
                            !the TOP LEVEL MENU to set up  the parameter values.  They cannot be!)
                           (clause_print !changed from this level.!))
                    (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (modified_on (!6_13_85!)))
(stage_1_rb_rule9 ($type (ifall))
                    ($if (triple (top_level_choice top_level_menu)
                                 =
                                 !Display outputs generated by simulation!)
                        ($not (triple (user simulation_run) = error_free)))
                    (created_by (samir))
                    (created_on (!4_29_85!))
                    ($then (clause_print
                            !The output files generated by the simulation program cannot be dis-!)
                           (clause_print
                            !played unless the FORTRAN program has run succesfully.  If you have!)
                           (clause_print
                            !not done so,  please use option 3  of the TOP LEVEL MENU to run the!)
                           (clause_print !simulation program.!))
                    (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (modified_on (!6_13_85!)))
(stage_1_rb_rule3 ($type (ifall))
                    (created_by (samir))
                    (created_on (!2_20_85!))
                    ($if (triple (top_level_choice top_level_menu)
                                 =
                                 !Display current parameter values required for simulation!)
                        (triple (initial_value frame_full) = true))
                    ($then (clause_print
                            !You will see a  menu of the parameters  which values can be seen by!)
                           (clause_print
                            !selecting  the appropiate choices.  You can return to the TOP LEVEL!)
                           (clause_print !MENU by using option 1.!)
                           (clause_print)
                           (frem 'disp_init_val_choice)
                           (loop 'disp))
                    (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (modified_on (!6_13_85!)))
(stage_1_rb_rule13 ($type (ifall))
                    ($if (triple (top_level_choice top_level_menu)
                                 =
                                 !Store current parameter values in a disk file!)
                         ($not (triple (initial_value frame_full) = true)))
                    ($then (clause_print
                            !There are no values  in the database,  hence no storage is possible!)
                           (clause_print
                            !at this point.  You are suggested to  use option 2 of the TOP LEVEL!)
                           (clause_print MENU.))
                    (created_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                    (created_on (!6_13_85!)))
(stage_1_rb_rule11 ($type (ifall))
                    (created_by (samir))
                    (created_on (!5_5_85!))
                    ($if (triple (top_level_choice top_level_menu)
                                 =
```

```
                        !Display current parameter values required for simulation!)
                    ($not (triple (initial_value frame_full) = true)))
                   ($then
                    (clause_print
                     !There are no values stored in the database, hence no display is possible.!))
                   (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                   (modified_on (!6_16_85!)))
(stage_1_rb_rule6 ($type (ifall))
                   (created_by (samir))
                   (created_on (!2_25_85!))
                   ($if
                    (triple (top_level_choice top_level_menu)
                            =
                            !Set up initial parameter values to default values!))
                   ($then (frem 'Y_matrix)
                          (frem 'quaternion)
                          (frem 'response_chosen)
                          (frem 'controller_type_desired)
                          (frem 'inertial_mat)
                          (frem 'Kd_matrix)
                          (frem 'Kp_matrix)
                          (frem 'Kd)
                          (frem 'initial_value)
                          (frem 'inertial_matrix)
                          (frem 'Kp)
                          (exec ls)
                          (clause_print)
                          (clause_print
                           !Just above is a listing of the current directory. Enter the name of the!)
                          (clause_print
                           !file which contains the default values.  Be sure the file exists in the!)
                          (clause_print
                           !current directory,  and it is a file created using  option 8 of the TOP!)
                          (clause_print
                           !LEVEL MENU.  If the file does not exist, then enter the name of any non!)
                          (clause_print !existing file.!)
                          (clause_print)
                          (msg "  >> ")
                          (set_up_frames_from_file (read)))
                   (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                   (modified_on (!6_16_85!)))
(stage_1_rb_rule4 ($type (ifall))
                   (created_by (samir))
                   (created_on (!2_25_85!))
                   ($if (triple (top_level_choice top_level_menu)
                                =
                                !Store current parameter values in a disk file!)
                        (triple (initial_value frame_full) = true))
                   ($then (exec ls)
                          (clause_print)
                          (clause_print
                           !Just above is a listing of all files in the current directory. Enter the!)
                          (clause_print
                           !name of the file where the current parameter values will be stored.   Be!)
                          (clause_print
                           !sure the name you enter does not conflict with the existing files in the!)
                          (clause_print !present directory.!)
                          (clause_print)
                          (msg "  >> ")
                          (storage_of_initial_values))
                   (modified_by (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
```

```
                    (modified_on (16_16_851)))
(change_param_control
 (prompt_specs (get_alternatives_from (param_menu))
               (put_data_in (change_param_choice))
               (data_input)))
(disp_init_val_control
 (prompt_specs (get_alternatives_from (param_menu))
               (put_data_in (disp_init_val_choice))
               (data_input)))
```

```
(param_menu (|Return to TOP LEVEL MENU|)
            (|Response type|)
            (|Axis of input command|)
            (TO)
            (Tfinal)
            (|DeltaT (time increment)|)
            (|Multi-step integration error|)
            (|Initial values of Omega and Theta|)
            (|Kp Matrix|)
            (|Kd Matrix|)
            (|Inertial Matrix|)
            (Quaternion)
            (|Amplitude of input wave signal|)
            (|Initial lowest frequency|)
            (|Frequency deltaT|)
            (|Number of decades|)
            (|Number of sampling frequencies/decade|)
            (|Phase of input wave signal|)
            (|Steady state error|))
```

```
(disp_init_val_agenda ((reset_rule_base 'disp_init_val_rb))
                      ((from 'disp_init_val_choice))
                      ((menu_input 'disp_init_val_control))
                      ((forward 'disp_init_val_rb)))
(disp_init_val_rb (if_removed (d_remove_rules))
                  (created_by (samir))
                  (created_on (12_25_85!))
                  (rules (disp_init_val_rb_rule1)
                         (disp_init_val_rb_rule2)
                         (disp_init_val_rb_rule3)
                         (disp_init_val_rb_rule4)
                         (disp_init_val_rb_rule5)
                         (disp_init_val_rb_rule6)
                         (disp_init_val_rb_rule7)
                         (disp_init_val_rb_rule8)
                         (disp_init_val_rb_rule9)
                         (disp_init_val_rb_rule10)
                         (disp_init_val_rb_rule11)
                         (disp_init_val_rb_rule12)
                         (disp_init_val_rb_rule13)
                         (disp_init_val_rb_rule17)
                         (disp_init_val_rb_rule18)
                         (disp_init_val_rb_rule14)
                         (disp_init_val_rb_rule15)
                         (disp_init_val_rb_rule16)
                         (disp_init_val_rb_rule19))
                  (params_in_ifs
                   (disp_init_val_choice
                    (param_menu
                     (rule (disp_init_val_rb_rule1)
                           (disp_init_val_rb_rule2)
                           (disp_init_val_rb_rule3)
                           (disp_init_val_rb_rule4)
```

```
                        (disp_init_val_rb_rule5)
                        (disp_init_val_rb_rule6)
                        (disp_init_val_rb_rule7)
                        (disp_init_val_rb_rule8)
                        (disp_init_val_rb_rule9)
                        (disp_init_val_rb_rule10)
                        (disp_init_val_rb_rule11)
                        (disp_init_val_rb_rule12)
                        (disp_init_val_rb_rule13)
                        (disp_init_val_rb_rule17)
                        (disp_init_val_rb_rule18)
                        (disp_init_val_rb_rule14)
                        (disp_init_val_rb_rule15)
                        (disp_init_val_rb_rule16)
                        (disp_init_val_rb_rule19))))))
(disp_init_val_rb_rule1 ($type (ifall))
                        ($if (triple (disp_init_val_choice param_menu) = T0))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($then (delim_display)
                               (fprint '(initial_value T0)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule2 ($type (ifall))
                        ($if
                         (triple (disp_init_val_choice param_menu) = Tfinal))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($then (delim_display)
                               (fprint '(initial_value Tfinal)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule3 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |DeltaT (time increment)|))
                        ($then (delim_display)
                               (fprint '(initial_value deltaT)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule4 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Multi-step integration error|))
                        ($then (delim_display)
                               (fprint '(initial_value error)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule5 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 Quaternion))
```

```
                        ($then (delim_display) (fprint '(quaternion)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule6 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Initial values of Omega and Theta|))
                        ($then (delim_display) (fprint 'Y_matrix))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule7 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_25_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Steady state error|))
                        ($then (delim_display)
                               (fprint
                                '(initial_value steady_state_error)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
```

```
(disp_init_val_rb_rule8 ($type (ifall))
                        (created_by (samir))
                        (created_on (12_27_85))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 !Frequency deltaT!))
                        ($then (delim_display)
                               (terpri)
                               (clause_print
                                !The inverse of frequncy deltaT is displayed below.!)
                               (fprint
                                '(initial_value frequency_deltaT)))
                        (modified_by (wants_to_run_simulation))
                        (modified_on (18_1_85)))
```

```
(disp_init_val_rb_rule9 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Amplitude of input wave signal|))
                        ($then (delim_display)
                               (fprint '(initial_value amplitude)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule10 ($type (ifall))
                         (created_by (samir))
                         (created_on (|2_27_85|))
                         ($if
                          (triple (disp_init_val_choice param_menu)
                                  =
                                  |Initial lowest frequency|))
                         ($then (delim_display)
                                (fprint
                                 '(initial_value init_frequency_value)))
                         (modified_by (|Juan J. Rodriguez-Moscoso|))
                         (modified_on (|6_15_85|)))
(disp_init_val_rb_rule11 ($type (ifall))
```

```
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Number of decades|))
                        ($then (delim_display)
                               (fprint
                                '(initial_value number_of_decades)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule12 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Number of sampling frequencies/decade|))
                        ($then (delim_display)
                               (fprint
                                '(initial_value
                                  number_of_sampling_frequency_per_decade)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule13 ($type (ifall))
                        (created_by (samir))
                        (created_on (|2_27_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Phase of input wave signal|))
                        ($then (delim_display)
                               (fprint '(initial_value phase)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule17 ($type (ifall))
                        (created_by (samir))
                        (created_on (|5_3_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Amplitude of impulse|))
                        ($then (delim_display)
                               (fprint
                                '(initial_value impulse_amplitude)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule18 ($type (ifall))
                        (created_by
                         (|Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh|))
                        (created_on (|6_10_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Response type|))
                        ($then (delim_display)
                               (fprint
                                '(response_chosen response_type)))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_on (|6_15_85|)))
(disp_init_val_rb_rule14 ($type (ifall))
```

```
                        (created_by (samir))
                        (created_on (|3_12_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Kp Matrix|))
                        ($then (delim_display)
                               (clause_print)
                               (clause_print
                                |The Kp matrix is being displayed:|)
                               (clause_print
                                ---------------------------------)
                               (printarray 'Kp)
                               (terpri))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_by (|6_15_85|)))
(disp_init_val_rb_rule15 ($type (ifall))
                        (created_by (samir))
                        (created_on (|3_12_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Kd Matrix|))
                        ($then (delim_display)
                               (clause_print)
                               (clause_print
                                |The Kd matrix is being displayed:|)
                               (clause_print
                                ---------------------------------)
                               (printarray 'Kd)
                               (terpri))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_by (|6_15_85|)))
(disp_init_val_rb_rule16 ($type (ifall))
                        (created_by (samir))
                        (created_on (|3_12_85|))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Inertial Matrix|))
                        ($then (delim_display)
                               (clause_print)
                               (clause_print
                                |The Inertial matrix is being displayed:|)
                               (clause_print
                                ---------------------------------------)
                               (printarray 'inertial_matrix)
                               (terpri))
                        (modified_by (|Juan J. Rodriguez-Moscoso|))
                        (modified_by (|6_15_85|)))
(disp_init_val_rb_rule19 ($type (ifall))
                        ($if
                         (triple (disp_init_val_choice param_menu)
                                 =
                                 |Axis of input command|))
                        (created_by (samir))
                        (created_on (|6_15_85|))
                        ($then (delim_display)
                               (clause_print)
                               (clause_print
                                |Displaying the axis of the input command|)
```

```
                        (fprint
                         '(initial_value axis_of_input_command)))
                    (modified_by (|Juan J. Rodriguez-Moscoso|))
                    (modified_on (|6_15_85|)))
(change_param_agenda ((reset_rule_base 'change_param_rb))
                     ((frem 'change_param_choice))
                     ((menu_input 'change_param_control))
                     ((forward 'change_param_rb)))
(output_display_menu (|Return to TOP LEVEL MENU|)
                     (|Plot of omega|)
                     (|Plot of theta|)
                     (|Characteristics of the step response analysis|)
                     (|Numerical output generated by simulation|))
(output_display_control (print
                         |The simulation program generates the following outputs:|)
                        (print |        - Options 2 and 3 are ASCII file plottings.|)
                        (print |        - Options 4 and 5 are numerical outputs.|)
                        (print
                         |Be sure to have run the simulation before observing the|)
                        (print
                         |outputs.  The outputs shown will correspond to the out-|)
                        (print
                         |puts generated when the simulation was last run.|)
                        (prompt_specs (get_alternatives_from
                                       (output_display_menu))
                                      (put_data_in (output_display_choice))
                                      (data_input)))
(output_display_agenda ((frem 'output_display_choice))
                       ((reset_rule_base 'output_display_rb))
                       ((menu_input 'output_display_control))
                       ((forward 'output_display_rb)))
(output_display_rb (if_removed (d_remove_rules))
                   (created_by (samir))
                   (created_on (|4_4_85|))
                   (rules (output_display_rb_rule4)
                          (output_display_rb_rule7)
                          (output_display_rb_rule8)
                          (output_display_rb_rule9)
                          (output_display_rb_rule10)
                          (output_display_rb_rule11)
                          (output_display_rb_rule1)
                          (output_display_rb_rule2)
                          (output_display_rb_rule5)
                          (output_display_rb_rule6)
                          (output_display_rb_rule3)
                          (output_display_rb_rule12))
                   (params_in_ifs (output_display_choice
                                   (output_display_menu
                                    (rule (output_display_rb_rule4)
                                          (output_display_rb_rule7)
                                          (output_display_rb_rule8)
                                          (output_display_rb_rule9)
                                          (output_display_rb_rule10)
                                          (output_display_rb_rule11)
                                          (output_display_rb_rule1)
                                          (output_display_rb_rule2)
                                          (output_display_rb_rule5)
                                          (output_display_rb_rule6)
                                          (output_display_rb_rule3)
                                          (output_display_rb_rule12))))
                                  (user
```

```
                                (last_response_run
                                 (rule (output_display_rb_rule4)
                                       (output_display_rb_rule7)
                                       (output_display_rb_rule8)
                                       (output_display_rb_rule9)
                                       (output_display_rb_rule10)
                                       (output_display_rb_rule11)
                                       (output_display_rb_rule1)
                                       (output_display_rb_rule2)
                                       (output_display_rb_rule5)
                                       (output_display_rb_rule6)
                                       (output_display_rb_rule3)
                                       (output_display_rb_rule12))))))
(output_display_rb_rule4 ($type (ifall))
                         (created_by (samir))
                         (created_on (|4_4_85|))
                         ($then (exec cat numer0out.stp))
                         ($if (triple (output_display_choice
                                       output_display_menu)
                                      =
                                      |Numerical output generated by simulation|)
                              (triple (user last_response_run)
                                      =
                                      |Step response|))
                         (modified_by
                          (|Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh|))
                         (modified_on (|6_9_85|)))
(output_display_rb_rule7 ($type (ifall))
                         ($then (exec cat theta0cha.imp))
                         (created_by (samir))
                         (created_on (|5_4_85|))
                         ($if (triple (output_display_choice
                                       output_display_menu)
                                      =
                                      |Characteristics of the analysis|)
                              (triple (user last_response_run)
                                      =
                                      |Impulse response|))
                         (modified_by
                          (|Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh|))
                         (modified_on (|6_9_85|)))
(output_display_rb_rule8 ($type (ifall))
                         ($if (triple (output_display_choice
                                       output_display_menu)
                                      =
                                      |Numerical output generated by simulation|)
                              (triple (user last_response_run)
                                      =
                                      |impulse response|))
                         ($then (exec cat numer0out.imp))
                         (created_by (samir))
                         (created_on (|5_4_85|))
                         (modified_by
                          (|Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh|))
                         (modified_on (|6_9_85|)))
(output_display_rb_rule9 ($type (ifall))
                         ($then (exec cat numer0out.frq))
                         (created_by (samir))
                         (created_on (|5_4_85|))
                         ($if (triple (output_display_choice
                                       output_display_menu)
```

```
                                =
                                !Numerical output generated by simulation!)
                           (triple (user last_response_run)
                                =
                                !Frequency response!))
                      (modified_by
                       (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                      (modified_on (!6_9_85!)))
(output_display_rb_rule10 ($type (ifall))
                      ($then (frequency_output_display 'theta))
                      (created_by (samir))
                      (created_on (!5_4_85!))
                      ($if (triple (output_display_choice
                                    output_display_menu)
                                =
                                !Plot of theta!)
                           (triple (user last_response_run)
                                =
                                !Frequency response!))
                      (modified_by
                       (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                      (modified_on (!6_9_85!)))
(output_display_rb_rule11 ($type (ifall))
                      ($then (frequency_output_display 'omega))
                      (created_by (samir))
                      (created_on (!5_4_85!))
                      ($if (triple (output_display_choice
                                    output_display_menu)
                                =
                                !Plot of omega!)
                           (triple (user last_response_run)
                                =
                                !Frequency response!))
                      (modified_by
                       (!Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh!))
                      (modified_on (!6_9_85!)))
(output_display_rb_rule1 ($type (ifall))
                      (created_by (samir))
                      (created_on (!4_4_85!))
                      ($if (triple (output_display_choice
                                    output_display_menu)
                                =
                                !Plot of omega!)
                           (triple (user last_response_run)
                                =
                                !Step response!))
                      ($then (exec cat omega0plt.stp) (clear_display))
                      (modified_by
                       (!Bor-Jau Hsieh & Juan J. Rodriguez-Moscoso!))
                      (modified_on (!6_13_85!)))
(output_display_rb_rule2 ($type (ifall))
                      (created_by (samir))
                      (created_on (!4_4_85!))
                      ($if (triple (output_display_choice
                                    output_display_menu)
                                =
                                !Plot of theta!)
                           (triple (user last_response_run)
                                =
                                !Step response!))
                      ($then (exec cat theta0plt.stp) (clear_display))
```

```
                        (modified_by
                         (!Bor-Jau Hsieh & Juan J. Rodriguez-Moscoso!))
                        (modified_on (!6_13_85!)))
(output_display_rb_rule5 ($type (ifall))
                        (created_by (samir))
                        (created_on (!5_4_85!))
                        ($if (triple (output_display_choice
                                      output_display_menu)
                                     =
                                     !Plot of omega!)
                             (triple (user last_response_run)
                                     =
                                     !Impulse response!)))
                        ($then (exec cat omegaOplt.imp) (clear_display))
                        (modified_by
                         (!Bor-Jau Hsieh & Juan J. Rodriguez-Moscoso!))
                        (modified_on (!6_13_85!)))
(output_display_rb_rule6 ($type (ifall))
                        (created_by (samir))
                        (created_on (!5_4_85!))
                        ($if (triple (output_display_choice
                                      output_display_menu)
                                     =
                                     !Plot of theta!)
                             (triple (user last_response_run)
                                     =
                                     !Impulse response!)))
                        ($then (exec cat thetaOplt.imp) (clear_display))
                        (modified_by
                         (!Bor-Jau Hsieh & Juan J. Rodriguez-Moscoso!))
                        (modified_on (!6_13_85!)))
(output_display_rb_rule3 ($type (ifall))
                        (created_by (samir))
                        (created_on (!4_4_85!))
                        ($then (exec cat thetaOcha.stp))
                        ($if (triple (output_display_choice
                                      output_display_menu)
                                     =
                                     !Characteristics of the step response analysis!)
                             (triple (user last_response_run)
                                     =
                                     !Step response!)))
                        (modified_by (!Bor-Jau Hsieh!))
                        (modified_on (!6_17_85!)))
(output_display_rb_rule12 ($type (ifall))
                         (created_by (wants_to_run_simulation))
                         (created_on (!5_5_85!))
                         ($then (clause_print
                                 !This option is for STEP response only, for FREQUENCY response!)
                                (clause_print
                                 !please use options 2, 3, and 5.!))
                         ($if (triple (output_display_choice
                                       output_display_menu)
                                      =
                                      !Characteristics of the step response analysis!)
                              (triple (user last_response_run)
                                      =
                                      !Frequency response!)))
                         (modified_by (wants_to_run_simulation))
                         (modified_on (!6_17_85!)))
(run_rb (if_removed (d_remove_rules))
```

```
        (created_by (samir))
        (created_on (|4_22_85|))
        (rules (run_rb_rule2) (run_rb_rule1))
        (params_in_ifs (response_chosen
                         (response_type (rule (run_rb_rule2) (run_rb_rule1))))
                       (user (likes_Tfinal_value (rule (run_rb_rule2)))
                             (Tfinal_value_found (rule (run_rb_rule1)))
                             (wants_to_run_simulation (rule (run_rb_rule1))))
                       (initial_value (frame_full (rule (run_rb_rule1)))))
        (param_specs (frame_full (find_strategy (fget))))
        (param_conclusions (user (Tfinal_value_found (rule (run_rb_rule2))))))
(run_rb_rule2 ($type (ifall))
              (created_by (samirt))
              (created_on (|4_28_85|))
              ($if (triple (response_chosen response_type) = |Step response|)
                   (prog nil
                         (frem '(initial_value Tfinal))
                         (fput '(initial_value Tfinal)
                               (quotient 5.0 (one_over_tau 0)))
                         (fprint '(initial_value Tfinal)))
                   (triple (user likes_Tfinal_value) = yes))
              ($then (conclude (user Tfinal_value_found) true))
              ($else (conclude (user Tfinal_value_found) true)
                     (frem '(initial_value Tfinal))
                     (clause_print
                      |Enter your desired value for 'final time' (Tfinal):|)
                     (msg N "  >> ")
                     (fput '(initial_value Tfinal) (read)))
              (modified_by (|Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh|))
              (modified_on (|6_13_85|)))
(run_rb_rule1 ($type (ifall))
              (created_by (samir))
              (created_on (|4_22_85|))
              ($if (triple (initial_value frame_full) = true)
                   ($or ($and (triple (response_chosen response_type)
                                      =
                                      |Step response|)
                              (triple (user Tfinal_value_found) = true))
                        ($and
                         (triple (response_chosen response_type)
                                 =
                                 |Frequency response|))
                        (triple (response_chosen response_type)
                                =
                                |Impulse response|))
                   (triple (user wants_to_run_simulation) = yes))
              ($then (clear_display)
                     (clause_print
                      |Initial parameter values are being put into 'input file'|)
                     (setup_init_val_in_simula.inp)
                     (clause_print)
                     (clause_print
                      |The 'simulation program' is now being run...(Be patient)|)
                     (start_sim))
              (modified_by (|Juan J. Rodriguez-Moscoso & Bor-Jau Hsieh|))
              (modified_on (|6_16_85|));;
(run_agenda ((frem 'eigr))
            ((frem 'eigi))
            ((frem 'one_over_tau))
            ((frem 'user))
            ((store_values_in_lisparray_from_frame 'Kp))
```

```
((store_values_in_lisparray_from_frame 'Kd))
((store_values_in_lisparray_from_frame 'inertial_matrix))
((load_eigen_values_in_frame))
((reset_rule_base 'run_rb))
((back '(run_rb_rule1) 'run_rb)))
```

```
(user_rb (if_removed (d_remove_rules))
         (created_by (samir))
         (created_on (|5_4_85|))
         (param_specs (decision_about_removing_values (value_type (literal))
                                                      (value_multiplicity
                                                       (single))
                                                      (value_required (yes))
                                                      (prompt_format
                                                       (menu_input))
                                                      (find_strategy (ask))
                                                      (legal_values (yes) (no))

; <<<<< start back on the left <<<<<
     (message_format (print |Do you wish to change the 'Response type' ?|)
                     (print
                      | - If you say YES, all the current parameter values will be saved|)
                     (print
                      |   and you will be prompted for a new selection of 'Response type'.|)
                     (print
                      | - If NO, the 'Response type'  will be saved  and all the current|)
                     (print |   parameter values will be removed.|)))
; >>>>> continue on the right >>>>>
)
                      (decision_about_removing_value))
         (rules (user_rb_rule1))
         (params_in_ifs
          (user (decision_about_removing_values (rule (user_rb_rule1))))))
(user_rb_rule1 (created_by (samir))
               (created_on (|5_4_85|))
               ($else (reset_rule_base 'value_input_rb)
                      (frem '(user decision_about_removing_values))
                      (frem 'response_chosen)
                      (back '(value_input_rb_rule1)
                            'value_input_rb))
               ($type (ifall))
               ($if (triple (user decision_about_removing_values) = no))
               ($then (frem '(user decision_about_removing_values))
                      (eval_agenda 'sim_expert_agenda))
               (modified_by (likes_Tfinal_value))
               (modified_on (|6_15_85|)))
```

Simulation Model Listing

```
c +-------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center        !
c !                        Hunstville, Alabama                        !
c !                              1985                                 !
c +-------------------------------------------------------------------+

c ********************************************************************
c *                                                                  *
c *                     Subroutine ANALYSIS                          *
c *                                                                  *
c ********************************************************************
c
c       This soubroutine will analyze:
c
c       1) step response:
c         . Percent of Maximum Overshoot
c         . Peak Time
c         . Rising Time (10% - 90%)
c         . Delay Time (0% - 50%)
c         . Settling Time (within SSE which was provided by the user)
c
c       2) Frequency response:
c          when system in steady-state, the routine will calcualte
c          the summations of sine and cosine series.
c
c       3) Impulse response:
c         . Peak Vaule
c         . Peak Time
c         . Settling Time (within Steady State Error)
c
c       Written by Bor-Jau Hsieh (Andy) (Apr-03-85)
c       Revised by Bor-Jau Hsieh (Andy) &
c                  Juan J. Rodriguez-Moscoso     (Jun-12-85)

        SUBROUTINE ANALYSIS ( ndim, t, Y, Therr, Omerr )

c-------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                   *
c-------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        COMMON / FLAG / FLAG(7)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)
        COMMON / STEP / PMO(3)          , RTini(3)      , RTend(3)      ,
     +                  DT(3)           , ST(3)         , PT(3)         ,
     +                  RT(3)           , RTimin(3)     , RTemin(3)     ,
     +                  DTmin(3)        , Thpeak(3)
        COMMON / FRQ1 / Amp             , Omega         , Phase         ,
     +                  ts
        COMMON / FRQ2 / THAMP(3)        , THFEE(3)      , THAOL(3)      ,
     +                  THPOL(3)        , THS(3)        , THC(3)        ,
     +                  X1(3)           , X2(3)         ,
     +                  OMAMP(3)        , OMFEE(3)      , OMAOL(3)      ,
     +                  OMPOL(3)        , OMS(3)        , OMC(3)        ,
     +                  X3(3)           , X4(3)         , XN
        COMMON / IMPU / ST_imp(3)       , PT_imp(3)     , Thpeak_imp(3)
        COMMON / PI   / PI
        COMMON / DAMP / damp_flag
```

```
      DIMENSION       Y(10)          , Therr(3)      , Omerr(3)

      INTEGER         damp_flag
      REAL*8          Kp      , Kd    , INMAT , INMATV
      LOGICAL*1       FLAG

c-------------------------------------------------------------------------
c*                      COMPUTATIAONS                                    *
c-------------------------------------------------------------------------

c ------------------   STEP ANALYSIS   -------------------------------c
      IF ( FLAG(1) )
     .   THEN
          I = 3
          IF ( FLAG(4) ) I = I + 4

c ***     Calculate Maximum Overshoot (Thpeak)
          DO K = 1, 3
             IF ( DABS( Y(K+I) ) .GT. Thpeak(K) )
     .           THEN
                 Thpeak(K) = Y(K+I)
                     PT(K) = t
             END IF
          END DO

c ***     If no damping, only PMO & PT are computed.
          IF ( damp_flag .EQ. 0 ) GOTO 5
          DO J= 1, 3

c ***        Calculate Rising Time (RT)
             IF ( DABS( DABS(Y(J+I)) - 0.1d0*Thcom(J) )
     +            .LT. RTimin(J) )
     .           THEN
                 RTimin(J) = DABS( DABS(Y(J+I)) - 0.1d0*Thcom(J) )
                 RTini(J) = t
             END IF
             IF ( DABS( Y(J+I) - 0.9d0*Thcom(J) )
     +            .LT. RTemin(J) )
     .           THEN
                 RTemin(J) = DABS(Y(J+I) - 0.9d0*Thcom(J))
                 RTend(J) = t
             END IF

c ***        Calculate Delay Time (DT)
             IF ( DABS(Y(J+I) - 0.5d0*Thcom(J))
     +            .LT. DTmin(J) )
     .           THEN
                 DTmin(J) = DABS(Y(J+I) - 0.5d0*Thcom(J))
                    DT(J) = t
             END IF

c ***        Calculate Settling Time (ST)
             IF ( Y(J+I) .NE. 0.0D0 ) THEN
                IF ( DABS(Y(J+I) - Thcom(J))
     +               .GT. (Bound(6)*Thcom(J)/100.d0) )
     +              THEN
                    ST(J) = 0.0D0
                   ELSE
                    IF ( ST(J) .EQ. 0.0D0 ) ST(J) = t
```

```
                  END IF
               END IF
            END DO
5           RETURN
         END IF

c ------------------   FREQUENCY ANALYSIS   --------------------------c
         IF ( FLAG(2) )
     +      THEN
             IF ( t .LT. ts ) GOTO 10

c ***        Integrate SINE and COSINE Series
             THS(1) = THS(1) + DSIN(Omega*t)*Therr(1)
             THS(2) = THS(2) + DSIN(Omega*t)*Therr(2)
             THS(3) = THS(3) + DSIN(Omega*t)*Therr(3)

             THC(1) = THC(1) + DCOS(Omega*t)*Therr(1)
             THC(2) = THC(2) + DCOS(Omega*t)*Therr(2)
             THC(3) = THC(3) + DCOS(Omega*t)*Therr(3)

             OMS(1) = OMS(1) + DSIN(Omega*t)*Omerr(1)
             OMS(2) = OMS(2) + DSIN(Omega*t)*Omerr(2)
             OMS(3) = OMS(3) + DSIN(Omega*t)*Omerr(3)

             OMC(1) = OMC(1) + DCOS(Omega*t)*Omerr(1)
             OMC(2) = OMC(2) + DCOS(Omega*t)*Omerr(2)
             OMC(3) = OMC(3) + DCOS(Omega*t)*Omerr(3)

c ***        Compute the new Angular Position & Rate errors, and
c            check Quaternion.
10           CONTINUE
             IF ( FLAG(4) ) THEN
                prototype = 1.0
               ELSE
                prototype = 0.0
             END IF

             Therr(1) = Therr(1) + Amp*DSIN(Omega*t+Phase)*Bound(13)
             Therr(2) = Therr(2) + Amp*DSIN(Omega*t+Phase)*Bound(14)
             Therr(3) = Therr(3) + Amp*DSIN(Omega*t+Phase)*Bound(15)

             Omerr(1) = Omerr(1) + Amp*Omega*DCOS(Omega*t+Phase)*
     +                   Bound(13)* (1.0-prototype)
             Omerr(2) = Omerr(2) + Amp*Omega*DCOS(Omega*t+Phase)*
     +                   Bound(14)* (1.0-prototype)
             Omerr(3) = Omerr(3) + Amp*Omega*DCOS(Omega*t+Phase)*
     +                   Bound(15)* (1.0-prototype)
             RETURN
         END IF

C --------------------  IMPULSE ANALYSIS  ----------------------------C
         IF ( FLAG(3) ) THEN
c ***       Checking out Quaternion Flag
            I = 3
            IF ( FLAG(4) ) I = I + 4
            DO J = 1, 3
c ***          Calculate Peak Value (Thpeak_imp)
               IF ( Y(J+I) .GT. Thpeak_imp(J) ) THEN
                  Thpeak_imp(J) = Y(J+I)
                      PT_imp(J) = t
```

```
                END IF

c ***           Calculate Settling Time (ST_imp)
                IF ( ABS(Y(J+I)) .GT. (Bound(6)/100.d0) ) THEN
                   ST_imp(J) = 0.d0
                  ELSE
                   IF (ST_imp(J) .EQ. 0.d0) THEN
                      ST_imp(J) = t
                   END IF
                END IF
             END DO
             RETURN
          END IF

          END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     SUBROUTINE BODY_DYNAM                          *
c *                                                                    *
c **********************************************************************

c ***   Routine written by Juan J. Rodriguez-Moscoso     (Jan-24-85)
c       Revised by Juan J. Rodriguez-Moscoso &
c                          Bor-Jau Hsieh (Andy)         (May-10-85)

        SUBROUTINE BODY_DYNAM ( ndim, Torque, Y, Ydot )

c--------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSIONS ARRAYS                  *
c--------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)
        COMMON / FLAG / FLAG(7)

        DIMENSION       Torque(3)       ,       ! Torque command.
     +                  Y(10)           ,       ! state vector.
     +                  Ydot(10)        ,       ! derivative of state
     +                  tmp(3), tmp1(3)         ! Local storages.

        REAL*8          Kp      , Kd    , INMAT , INMATV
        LOGICAL*1       FLAG

c--------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c--------------------------------------------------------------------
c ***   BODY DYNAMICS   ***

        DO i=1,3
                ia = i*(11-3*i)/2-2
                ib = i*(3*i-13)/2+8
            tmp(i) = 0.0d0
           tmp1(i) = 0.0d0

           DO j=1,3
               tmp(i) = tmp(i) + Y(j)*INMAT(ib,j)
              tmp1(i) = tmp1(i) + Y(j)*INMAT(ia,j)
           END DO

            tmp(i) = Y(ia)*tmp(i)
           tmp1(i) = Y(ib)*tmp1(i)
            tmp(i) = Torque(i) - tmp(i) + tmp1(i)
        END DO

        DO i=1,3
           Ydot(i) = 0.d0
           DO k=1,3
              Ydot(i) = Ydot(i) + INMATV(i,k)*tmp(k)
```

```
            END DO
          END DO

c ***   Quaternion check out. If the Quaternion block is present during
c       the SIMULATION, then RETURN and let the QUATERNION routine make
c       the computational work to forming the quaternion-rate eqs.

          IF ( FLAG(4) ) RETURN

          Ydot( 4) = Y( 1)
          Ydot( 5) = Y( 2)
          Ydot( 6) = Y( 3)

          RETURN
          END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     Subroutine CHECK_ERROR                         *
c *                                                                    *
c **********************************************************************

c ***   This subroutine check error flag "ihlf" for both
c       Runge-Kutta and Predictor-Corrector integration methods.
c
c ***   Written by Bor-Jau Hsieh (Andy) &
c                  Juan J. Rodriguez-Moscoso (May-12-85)

        SUBROUTINE CHECK_ERROR ( ihlf, ICODE )

c---------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSIONS ARRAYS                   *
c---------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON / FLAG / FLAG(7)
        LOGICAL*1       FLAG

c---------------------------------------------------------------------
c*                        COMPUTATIONS                               *
c---------------------------------------------------------------------
        ICODE = 0
         ivar = ihlf - 10
        IF ( ivar ) 50, 50, 10
10      GO TO ( 20, 30, 40 ), ivar

20      CONTINUE
c ***   Check FLAG(i) [i=1,2,3] to determine OUTPUT unit
        DO i=1,3
           IF ( FLAG(i) )
              THEN
               WRITE(i,25)
           END IF
        END DO
        ICODE = 1
        RETURN

30      CONTINUE
c ***   Check FLAG(i) [i=1,2,3] to determine OUTPUT unit
        DO i=1,3
           IF ( FLAG(i) )
              THEN
               WRITE(i,35)
           END IF
        END DO
        ICODE = 1
        RETURN

40      CONTINUE
c ***   Check FLAG(i) [i=1,2,3] to determine OUTPUT unit
        DO i=1,3
```

```
            IF ( FLAG(i) )
               THEN
                WRITE(i,45)
            END IF
         END DO
         ICODE = 1
         RETURN

50       RETURN

c======================== FORMATS ====================================c
25       FORMAT ( 1HO,15X,
     +           '=================== ERROR  DETECTED ==================='
     +           /1H ,15X,
     +           '  There was no Convergence by applying this method of  '
     +           /1H ,15X,
     +           '  Integration. Execution stopped.')
35       FORMAT ( 1HO,15X,
     +           '=================== ERROR  DETECTED ==================='
     +           /1H ,15X,
     +           ' Both, initial and final values of the interval under  '
     +           /1H ,15X,
     +           ' consideration, are zero. Execution stopped.')
45       FORMAT ( 1HO,15X,
     +           '=================== ERROR  DETECTED ==================='
     +           /1H ,15X,
     +           'The final value of the interval under consideration is '
     +           /1H ,15X,
     +           'less than the initial value. Execution Stopped.')
c=============================================================================c
         END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                      Subroutine CONTROLLER                         *
c *                                                                    *
c **********************************************************************

c       External subroutine to compute the right hand sides Ydot of the
c       system to given values x and Y. This routine, if accessed,
c       should not destroy x and Y.
c
c       For this particular application, we consider Ydot as the
c       derivative of omega and theta in THE BOdy DyNamics equations.

c ***   Routine written by Juan J. Rodriguez-Moscoso (Jan-24-85)
c          Last revised by Juan J. Rodriguez-Moscoso
c                          Bor-Jau Hsieh (Andy)      (May-05-85)

        SUBROUTINE CONTROLLER ( ndim, Therr, Omerr, Torque )

c---------------------------------------------------------------------
c*              COMMON AREAS DEFINITIONS AND ARRAYS                   *
c---------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)

        DIMENSION       Torque(3)       , Therr(3)      , Omerr(3)
        REAL*8          Kp      , Kd    , INMAT , INMATV

c---------------------------------------------------------------------
c*                      COMPUTATIONS                                  *
c---------------------------------------------------------------------
c ***   The torque dimensioning will be 3 [= ndim/2]. Its computation is
c       done in this subroutine, but it can be modified to another rou-
c       tine which may be called from this one; i.e, the torque calcula-
c       tions will be enclosed in this block of programing.

c ***   Also, the Input commands (desired output) is passed thru the
c       COMMON statement:
c
c       COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
c    .                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
c    .                  Thcom(3)        , Omcom(3)
c       where,
c                Bound: Boundary conditions array.
c                  Yin: Vector of states.
c                   Kp: Matrix Kp(i,j) Controller,
c                   Kd: Matrix Kd(i,j) Controller,
c                INMAT: Inertia Matrix, and
c               INMATV: Inverse Inertia Matrix.
c                Thcom: Position Command,
c                Omcom: Angular Velocity Command,
c       and,
```

```
c                Therr: Vehicle Position Error,
c                Omerr: Vehicle Rate Error.

c ***   Computing Torque(i).    ***

        DO i=1,3
           Torque(i) = 0.0D0
           DO j=1,3
              Torque(i) = Torque(i) + Kp(i,j)*Therr(j) +
     +                    Kd(i,j)*Omerr(j)
           END DO
        END DO

        RETURN
        END
```

```
c ***************************************************************************
c *                                                                         *
c *                      SUBROUTINE DRAW_AXES                               *
c *                                                                         *
c ***************************************************************************

c.....  Written by Juan J. Rodriguez-Moscoso on 22-May-85
c       Last revised on 22-Jun-85

c.....  This routine draws the vertical and horizontal axes of the screen
c.....  plot generated by SCR_PLOTTER.

        SUBROUTINE DRAW_AXES ( iunit, Ymax, Ymin, t, nt )

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        COMMON / FLAG / FLAG(7)
        COMMON / SCRP / number_records,
     .                  number_decades,
     .                  number_samp_f

        DIMENSION       t(nt)
        INTEGER         cero_pos
        LOGICAL*1       FLAG
        CHARACTER*5     zero_pos, mover10
        CHARACTER*4     up1
        CHARACTER*3     home
        CHARACTER*1     BAR, PLUS, MENOS, blank
        CHARACTER*61    AXIS

        DATA BAR/'!'/, PLUS/'+'/, MENOS/'-'/, blank/' '/

        call cursor_home( home, 1, idummy )
        call cursor_up( up1, 1, idum, 1 )
        WRITE(iunit,*) home
        IF ( Ymax .EQ. 0.0 .AND. Ymin .EQ. 0.0 )
     .     THEN
            Ymax = 1.0
            Ymin = -1.0
        END IF

        WRITE(iunit,*) home
        call cursor_right( mover10, 1, idummy, 10 )
        do i=1,21
           Y = Ymax - (i-1)*(Ymax-Ymin)/20
           WRITE(iunit,1010) Y, BAR
        end do

        IF ( FLAG(2) )          ! Only for FREQUENCY Plotting.
     .     THEN
            DO i=1,number_decades-1
               muevete = i*60/number_decades + 10
               call cursor_right( zero_pos, 1, idummy, muevete )
               WRITE(iunit,*) home
               DO j=1,21
                  WRITE(iunit,*) zero_pos, BAR
               END DO
            END DO
        END IF
```

```
      call cursor_right( zero_pos, 1, idummy, 70 )
      WRITE(iunit,*) home
      do i=1,21
         WRITE(iunit,*) zero_pos, BAR
      end do

      IF ( FLAG(1) )
        THEN
         do i=1,6
            ii = 10*i-9
            AXIS(ii:ii) = PLUS
            do j=1,9
               ij = ii + j
               AXIS(ij:ij) = MENOS
            end do
         end do
        ELSE
         DO J=1,61
            AXIS(J:J) = MENOS
         END DO
         AXIS(1:1) = PLUS
         do j=1,number_decades-1
            ntemp = j*60/number_decades + 1
            AXIS(ntemp:ntemp) = PLUS
        end do
      END IF
      AXIS(61:61) = PLUS
      WRITE(iunit,*) up1, mover10, AXIS

      IF ( FLAG(2) )
         THEN
          GO TO ( 10, 20, 30, 40, 50, 60 ), number_decades
10        WRITE(iunit,1030) t, up1
          GO TO 70
20        WRITE(iunit,1040) t, up1
          GO TO 70
30        WRITE(iunit,1050) t, up1
          GO TO 70
40        WRITE(iunit,1060) t, up1
          GO TO 70
50        WRITE(iunit,1070) t, up1
          GO TO 70
60        WRITE(iunit,1020) t, up1
          GO TO 70
         ELSE
          WRITE(iunit,1020) t, up1
      END IF

70    CONTINUE
      IF ( Ymin .GT. 0.0D0 )
         THEN
          cero_pos = 1        !0
         ELSE
              cero = - 20*Ymin/( Ymax - Ymin ) + 1.0D0
          cero_pos = NINT( cero )
      END IF
      zero_pos(1:5) = blank
      call cursor_up( zero_pos, 1, idummy, cero_pos )
      IF ( cero_pos .NE. 0 )
         THEN
```

```
            ipos = idummy - 1
            WRITE(iunit,*) zero_pos(1:ipos), mover10, AXIS, up1
        END IF
        WRITE(iunit,*) home
        WRITE(iunit,*) mover10, AXIS

        RETURN

C=======================  FORMATS  ====================================C
1010    FORMAT(1H ,G9.3,1X,A)
1020    FORMAT(1H ,7X,7(G10.2),A)
1030    FORMAT(1H ,7X,G10.2,50X,G10.2,A)
1040    FORMAT(1H ,7X,2(G10.2,20X),G10.2,A)
1050    FORMAT(1H ,7X,3(G10.2,10X),G10.2,A)
1060    FORMAT(1H ,7X,4(G10.2,5X),G10.2,A)
1070    FORMAT(1H ,7X,5(G10.2,2X),G10.2,A)
        END
```

```
c ------------------------------------------------------------------------
c !             Copyright (c) NASA Marshall Space Flight Center          !
c !                        Hunstville, Alabama                           !
c !                              1985                                    !
c ------------------------------------------------------------------------

c ************************************************************************
c *                                                                      *
c *                       SUBROUTINE EIGEN                               *
c *                                                                      *
c ************************************************************************

c ***   Subroutine written by Juan J. Rodriguez-Moscoso (Apr-18-85)

c       Revised on Apr-19-85    ! Includding computation of A matrix,
c                                 where the A matrix is fixed by Kp,
c                                 Kd, and the Inertia matrices. Also,
c                                 it is given that INMAT = a*Identity.
c               on May-09-85    ! Commenting out first approach of the
c                                 eigenvalues computation.

        SUBROUTINE EIGEN( KP, KD, INMAT, TAU, EIGR, EIGI, IERR )

c ***   If IERR = 0     , then Eigenvalues Computation is successful.
c          IERR !=0     , then an ERROR has been produced and no sol-
c                         ution has been reached.

c------------------------------------------------------------------------
c*                       ARRAY DIMENSIONING                             *
c------------------------------------------------------------------------

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        REAL*8          KP, KD, INMAT, INMATV
        DIMENSION       KP(3,3), KD(3,3), INMAT(3,3), EIGR(6), EIGI(6),
     +                  A(6,6), INMATV(3,3), INT(6), SCALE(6)
        DATA A/36*0.D0/

c ***   Finding the inverse of the Inertia matrix

        DO 5 I=1,3
           DO 4 J=1,3
              INMATV(I,J) = INMAT(I,J)
4          CONTINUE
5       CONTINUE

        CALL MATINV( INMATV,3,ICODE )

c ***   If ICODE is not zero ==> The Inertia matrix is singular.

        IF( ICODE .NE. 0 ) THEN
                IERR = -1               ! IERR = -1 No inverse
                RETURN
        END IF

c ***   Forming the A matrix where
c                    ! - Kd*INMATV  - Kp*INMATV !
c               A = ! ..........................!
c                    !   Identity         0     !
```

```
      DO 20 I=1,3
         DO 10 J=1,3
            A(I,J) = 0.DO
            DO 10 K=1,3
               A(I,J) = A(I,J)-INMATV(I,K)*KD(K,J)
10          CONTINUE
            DO 20 J=1,3
               JI = J+3
               DO 20 K=1,3
                  A(I,JI) = A(I,JI)-INMATV(I,K)*KP(K,J)
20    CONTINUE
      DO 30 I=4,6
         A(I,I-3) = 1.DO
30    CONTINUE

c ***   Computing the Eigenvalues of A

      CALL BALANC( 6, 6, A, LOW, IGH, SCALE )
      CALL ELMHES( 6, 6, LOW, IGH, A, INT )
      CALL HQR( 6, 6, LOW, IGH, A, EIGR, EIGI, IERR )

c ***   End of Eigenvalues Computation

c ***   Computing the maximum eigenvalue in module. We assume that
c       real parts of the eigenvalues are less than zero.

      TAU = DABS( EIGR(1) )
      DO 40 I=2,6
         IF ( TAU .LT. DABS(EIGR(I)) ) TAU = DABS( EIGR(I) )
40    CONTINUE

      RETURN
      END

c *************************************************************************
c *                                                                       *
C *                     SUBROUTINE BALANC                                 *
c *                                                                       *
C *************************************************************************

c ***   Routine written by Juan J. Rodriguez-Moscoso (Apr-3-85)

c       Revised by Juan J. Rodriguez-Moscoso (Apr-18-85)

      SUBROUTINE BALANC( NM,N,A,LOW,IGH,SCALE )

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION       A(NM,N), SCALE(N)
      LOGICAL         NOCONV

c ***   RADIX specifies the Base of the machine floating point representation
      RADIX = 2.DO
         B2 = RADIX*RADIX
          K = 1
          L = N
      GOTO 100

c ***   Exchanging row and column

20    SCALE(M) = J
```

```
      IF( J .NE. M ) THEN
         DO 30 I=1,L
               F = A(I,J)
            A(I,J) = A(I,M)
            A(I,M) = F
30       CONTINUE

         DO 40 I=K,N
               F = A(J,I)
            A(J,I) = A(M,I)
            A(M,I) = F
40       CONTINUE
      END IF

      GOTO( 80,130), IEXC
c ***  Searching for rows isolating an eigenvalue and
c      push them down

80    IF( L .EQ. 1 ) THEN
         LOW = K
         IGH = L
         RETURN
      END IF

      L = L-1
100   DO 120 J=L,1,-1
         DO 110 I=1,L
            IF( I .NE. J ) THEN
               IF( A(J,I) .NE. 0.DO ) GOTO 120
            END IF
110      CONTINUE
            M = L
         IEXC = 1
         GOTO 20
120   CONTINUE
      GOTO 140

130   K = K+1
140   DO 170 J=K,L

         DO 150 I=K,L
            IF( I .NE. J ) THEN
               IF( A(I,J) .NE. 0.DO ) GOTO 170
            END IF
150      CONTINUE

         IEXC = 2
            M = K
         GOTO 20
170   CONTINUE

C ***  Balancing the submatrix in rows K to L

      DO 180  I=K,L
180      SCALE(I) = 1.DO

C ***  Iteration for Norm Reduction

190   NOCONV = .FALSE.
```

```
            DO 270 I=K,L
               C = 0.D0
               R = 0.D0

               DO 200 J=K,L
                  IF( J .NE. I ) THEN
                     C = C+DABS( A(J,I) )
                     R = R+DABS( A(I,J) )
                  END IF
200            CONTINUE

               G = R/RADIX
               F = 1.D0
               S = C + R

210            IF( C .LT. G ) THEN
                  F = F*RADIX
                  C = C*B2
                  GOTO 210
               END IF
               G = R*RADIX

230            IF( C .GE. G ) THEN
                  F = F/RADIX
                  C = C/B2
                  GOTO 230
               END IF

C ***       Balancing

               IF( (C+R)/F .LT. 0.95D0*S ) THEN
                  SCALE(I) = SCALE(I)*F
                         G = 1.D0/F
                    NOCONV = .TRUE.

                  DO 250 J=K,N
                     A(I,J) = A(I,J)*G
250               CONTINUE
                  DO 260 J=1,L
                     A(J,I) = A(J,I)*F
260               CONTINUE
               END IF

270         CONTINUE

            IF( NOCONV ) GOTO 190
            LOW = K
            IGH = L
            RETURN
            END
c ***************************************************************************
c *                                                                         *
c *                        SUBROUTINE ELMHES                                *
c *                                                                         *
c ***************************************************************************

            SUBROUTINE ELMHES( NM, N, LOW, IGH, A, INT )

c ***       Routine written by J. Rodriguez-Moscoso (Apr-03-85)
c           Last Revision: Apr-18-85 (JR-M)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION       A(NM,N), INT(IGH)

       LA = IGH-1
      KP1 = LOW+1
      IF( LA .LT. KP1 ) RETURN

      DO 180 M=KP1,LA
         MM1 = M-1
           X = 0.DO
           I = M

         DO 100 J=M,IGH
            IF( DABS(A(J,MM1)) .GT. DABS(X) ) THEN
               X = A(J,MM1)
               I = J
            END IF
100      CONTINUE

         INT(M) = I
         IF( I .NE. M ) THEN
C ***    Interchanging Rows and Columns of A   ***********
            DO 110 J=MM1,N
                  Y = A(I,J)
               A(I,J) = A(M,J)
               A(M,J) = Y
110         CONTINUE
            DO 120 J=1,IGH
                  Y = A(J,I)
               A(J,I) = A(J,M)
               A(J,M) = Y
120         CONTINUE
         END IF
C ***    End of the Interchange   ***********************

         IF( X .NE. 0.DO ) THEN
            MP1 = M+1
            DO 160 I=MP1,IGH
               Y = A(I,MM1)
               IF( Y .NE. 0.DO ) THEN
                        Y = Y/X
                  A(I,MM1) = Y
                  DO 140 J=M,N
                     A(I,J) = A(I,J)-Y*A(M,J)
140               CONTINUE
                  DO 150 J=1,IGH
                     A(J,M) = A(J,M)+Y*A(J,I)
150               CONTINUE
               END IF

160         CONTINUE
         END IF
180   CONTINUE

      RETURN
      END

c *********************************************************************
c *                                                                   *
```

```
c *                       SUBROUTINE HQR                                 *
c *                                                                      *
c ************************************************************************

c ***   Routine written by J. Rodriguez-Moscoso (Apr-02-85)
c               Revised by J. Rodriguez-Moscoso (Apr-18-85)

        SUBROUTINE HQR( NM,N,LOW,IGH,H,WR,WI,IERR )

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        REAL*8      MACHEP
        INTEGER     EN, ENM2
        DIMENSION   H(NM,N), WR(N), WI(N)
        LOGICAL     NOTLAS

C ***   MACHEP specifies precision of floating point arithmetic
c       MACHEP = 1.DO/2.DO**55

        MACHEP = .28D-16
          IERR = 0

C ***   Storing Roots isolated by BALANC

        DO 50 I=1,N
           IF( I .LT. LOW .OR. I .GT. IGH ) THEN
                        WR(I) = H(I,I)
                        WI(I) = 0.DO
           END IF
50      CONTINUE

        EN = IGH
         T = 0.DO

C ***   Searching for next Eigenvalues

60      IF( EN .LT. LOW ) RETURN
         ITS = 0
          NA = EN-1
        ENM2 = NA-1

C ***   Looking for single small sub-diagonal element

70      DO 80 L = EN,LOW,-1
           IF( L .EQ. LOW ) GOTO 100
           RINTER = MACHEP*( DABS(H(L-1,L-1)) + DABS(H(L,L)) )
           IF( DABS(H(L,L-1)) .LE. RINTER ) GOTO 100
80      CONTINUE

C ***   Forming shift

100     X = H(EN,EN)
        IF( L .EQ. EN ) GOTO 270
        Y = H(NA,NA)
        W = H(EN,NA)*H(NA,EN)
        IF( L .EQ. NA ) GOTO 280

        IF( ITS .EQ. 30 ) THEN
           IERR = EN
           RETURN
                        ELSE
```

```
            IF( ITS .NE. 10 .AND.
     $           ITS .NE. 20 ) GOTO 130
            T = T+X
            DO 120 I=LOW,EN
               H(I,I) = H(I,I)-X
120         CONTINUE

            S = DABS(H(EN,NA))+DABS(H(NA,ENM2))
            X = 0.75D0*S
            Y = X
            W = -0.4375D0*S*S
         END IF

130      ITS = ITS+1

C ***    Looking for two consecutive small sub-diagonal elements

         DO 140 M = ENM2,L,-1
            ZZ = H(M,M)
             R = X-ZZ
             S = Y-ZZ
             P = ( R*S-W )/H(M+1,M)+H(M,M+1)
             Q = H(M+1,M+1)-ZZ-R-S
             R = H(M+2,M+1)
             S = DABS(P)+DABS(Q)+DABS(R)
             P = P/S
             Q = Q/S
             R = R/S
            IF( M .EQ. L ) GOTO 150
            IF( DABS(H(M,M-1))*( DABS(Q)+DABS(R) ) .LE.
     $          MACHEP*DABS(P)*( DABS(H(M-1,M-1) )+DABS(ZZ)+
     $          DABS(H(M+1,M+1)) ) ) GOTO 150
140      CONTINUE

150      MP2 = M+2
         DO 160 I=MP2,EN
            H(I,I-2) = 0.D0
            IF( I .NE. MP2 ) H(I,I-3) = 0.D0
160      CONTINUE

C ***    Doubling QR step involving rows L to EN and columns M to EN

         DO 260 K=M,NA
            NOTLAS = .FALSE.
            IF( K .NE. NA ) NOTLAS = .TRUE.
            IF( K .NE. M ) THEN
                 P = H(K,K-1)
                 Q = H(K+1,K-1)
                 R = 0.D0
                 IF( NOTLAS ) R = H(K+2,K-1)
                 X = DABS(P)+DABS(Q)+DABS(R)
                 IF( X .EQ. 0.D0) GOTO 260
                 P = P/X
                 Q = Q/X
                 R = R/X
            END IF

c           S = SIGN( DSQRT(P*P+Q*Q+R*R),P )
c--->    The next change has been introduced in order to get
c        a compiled version of the routine under EUNICE F77
```

```
c       compiler <-------------------------------------------------------

           S = DSQRT( P*P+Q*Q+R*R )
           IF( P .LT. 0.D0 ) S = -1.D0*S

           IF( K .NE. M ) THEN
                H(K,K-1) = -S*X
                           ELSE
                IF( L .NE. M ) H(K,K-1) = -H(K,K-1)
           END IF

            P = P+S
            X = P/S
            Y = Q/S
           ZZ = R/S
            Q = Q/P
            R = R/P

C ***   Row Modification

           DO 210 J=K,EN
              P = H(K,J)+Q*H(K+1,J)
              IF( NOTLAS ) THEN
                         P = P+R*H(K+2,J)
                  H(K+2,J) = H(K+2,J)-P*ZZ
              END IF
              H(K+1,J) = H(K+1,J)-P*Y
                H(K,J) = H(K,J)-P*X
210        CONTINUE

           J = MINO( EN,K+3 )

C ***   Column Modification

           DO 230 I=L,J
              P = X*H(I,K)+Y*H(I,K+1)
              IF( NOTLAS ) THEN
                  P = P+ZZ*H(I,K+2)
                  H(I,K+2) = H(I,K+2)-P*R
              END IF
              H(I,K+1) = H(I,K+1)-P*Q
                H(I,K) = H(I,K)-P
230        CONTINUE
260     CONTINUE
        GOTO 70

C ***   One root found

270     WR(EN) = X+T
        WI(EN) = 0.D0
            EN = NA
        GOTO 60

C ***   Two roots found

280      P = ( Y-X )/2.D0
         Q = P*P+W
        ZZ = DSQRT( DABS(Q) )
         X = X+T
```

```
            IF ( Q .GE. 0.D0 ) THEN

C ***   Real Pair

C                   ZZ = P+SIGN( ZZ,P )
c -->   This change is introduced in order to get a compiled
c       version of the routine under EUNICE F77 compiler <---------------
                    IF( P .LT. 0.D0 ) ZZ = P+ZZ
                    IF( P .GE. 0.D0 ) ZZ = P-ZZ

                    WR(NA) = X+ZZ
                    WR(EN) = WR(NA)
                    IF( ZZ .NE. 0.D0 ) WR(EN) = X-W/ZZ
                    WI(NA) = 0.D0
                    WI(EN) = 0.D0
                                ELSE
C ***   Complex Pair

                    WR(NA) = X+P
                    WR(EN) = X+P
                    WI(NA) = ZZ
                    WI(EN) = -ZZ
            END IF

            EN = ENM2
            GOTO 60

            END
```

```
c +--------------------------------------------------------------------+
c |            Copyright (c) NASA Marshall Space Flight Center         |
c |                        Hunstville, Alabama                         |
c |                              1985                                  |
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                      Subroutine EULER                              *
c *                                                                    *
c **********************************************************************

c ***   This routine uses the Euler method for solving a system of
c       differential equations.

c ***   Written by Juan J. Rodriguez-Moscoso &
c                  Bor-Jau Hsieh                    (May-12-85)
c       Revised on Jun-02-85    : Restructure of Program.

        SUBROUTINE EULER ( ndim, t0 , tf, deltat, Y, Ydot )

c---------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                     *
c---------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)

        DIMENSION       Y(10)           , Ydot(10)
        REAL*8          Kp      , Kd    , INMAT , INMATV

c---------------------------------------------------------------------
c*                      COMPUTATIONS                                  *
c---------------------------------------------------------------------
        DO 100 t=t0,tf,deltat
           CALL SYS_DYNAM ( ndim, t, Y, Ydot )
           CALL OUTPUT_17 ( ndim, t, deltat, Y, Ydot )
           DO i=1,ndim
              Y(i) = Y(i) + deltat*Ydot(i)
           END DO
100     CONTINUE

        RETURN
        END
```

```
c ------------------------------------------------------------------------
c !             Copyright'(c) NASA Marshall Space Flight Center          !
c !                          Hunstville, Alabama                         !
c !                                1985                                  !
c ------------------------------------------------------------------------

c ************************************************************************
c *                                                                      *
c *              Subroutines for Graphic Purposes                        *
c *                                                                      *
c ************************************************************************

c ***   Written by Juan J. Rodriguez-Moscoso (Apr-21-85)

c       Revised on (May-05-85)  ! Adding unit as an argument to each
c                               ! routine in order to use them on file
c                               ! writing.
c       Revised on (May-07-85)  ! Including comments for each subroutine,
c                               ! and addition of new subroutines
c       Revised on (May-10-85)  ! Including new subroutines and
c                                 modification of the main routines.
c                on May-13-85   ! Adding type of format (formatted or un-
c                                 formatted) on the writing routines.

c       SUBROUTINES:                          PURPOSE:
c       ============                          ========
c
c       - Enter_graph_mode( iunit )     : This routine sets graphic mode.
c       - Exit_graph_mode( iunit )      : This routine resets normal
c                                         mode of operation.
c       - Enter_hold_screen_mode( iu )  : This routine sets the terminal
c                                         to be hold.
c       - Exit_hold_screen_mode( iu )   : This routine resets the termi-
c                                         nal to normal mode.
c       - Clear_display( iunit, ans )   : Clears screen.
c       - Transmit_page( iunit, ans )   : Transmits new page to the out-
c                                         put screen or unit 'iunit'.
c       - Transmit_curr_line( iunit )   : Transmits the current line of
c                                         output to the screen (iunit=6)
c                                         or unit 'iunit'.
c       - Transmit_char_at_cursor( iu ) : Transmits the current character
c                                         at the cursor to output screen
c                                         on unit 'iu'.
c       - Transmit_25th_line( iu )      : Transmits the 25th line to out-
c                                         put screen on unit 'iu'.
c       - Cursor_up( str, is, ie, n )   : Moves the cursor forward n pos-
c                                         itions. Results are written in
c                                         string 'str'.
c       - Cursor_down( str, is, ie, n ) : Moves the cursor backwards n
c                                         positions and result is stored
c                                         in string 'str'
c       - Cursor_right( s, is, ie, n )  : Moves the cursor to the right
c                                         n positions and result is
c                                         stored in string 's'.
c       - Cursor_left( s, is, ie, n )   : Moves the cursor to the left
c                                         n positions.
c       - Cursor_home( iunit )          : Moves the cursor to home posit.
c       - Enter_alt_char_set( iunit )   : Sets the alternate character
c                                         set of instructions.
c       - Exit_alt_char_set( iunit )    : Exits the alternate char. set.
```

```
c       - Enter_alt_char_graph_set( iu ): Enter the alternate character
c                                         graphic set of instructions.
c       - Exit_alt_char_graph_set( iu ) : Disables the action of the
c                                         above routine.
c       - Cursor_off( iunit )           : Turns off the cursor.
c       - Cursor_on( iunit )            : Turns on the cursor.
c       - Save_cursor_position( iunit ) : Saves the current cursor pos-
c                                         ition.
c       - Cursor_to_save_position( iu ) : Sets the cursor to the previous-
c                                         ly saved position.

c       *** WARNING ***
c       ------> Do not modify these routines without
c               talking to Juan J. Rodriguez-Moscoso <-------------------

        subroutine enter_graph_mode ( el_codigo, is, ie ) ! (May-10)
c ---   <ESC>[10m
        character*(*) el_codigo
        el_codigo(is:is+4) = char(27)//char(91)//char(49)//char(48)
     +                       //char(109)
        ie = is + 5
        return
        end
c
        subroutine exit_graph_mode ( el_codigo, is, ie )  ! (May-10)
c ---   <ESC>[11m
        character*(*) el_codigo
        el_codigo(is:is+4) = char(27)//char(91)//char(49)//char(49)
     +                       //char(109)
        ie = is + 5
        return
        end
c
        subroutine enter_hold_screen_mode ( el_codigo, is, ie )
c ---   <ESC>[>3h
        character*(*) el_codigo
        el_codigo(is:is+4) = char(27)//char(91)//char(62)//char(51)
     +                       //char(104)
        ie = is + 5
        return
        end
c
        subroutine exit_hold_screen_mode ( el_codigo, is, ie )
c ---   <ESC>[>3l
        character*(*) el_codigo
        el_codigo(is:is+4) = char(27)//char(91)//char(62)//char(51)
     +                       //char(108)
        ie = is + 5
        return
        end

        subroutine clear_display ( iunit, ans ) ! (Adding unit number)
c ---   <ESC>[2J<ESC>[H
        character*1     ans
        character*7     el_codigo
        el_codigo(1:3) = char(27)//char(91)//char(72)
        el_codigo(4:7) = char(27)//char(91)//char(50)//char(74)
        if ( ans .eq. '*' ) then
           write(iunit,*) el_codigo
           else
```

```
            write(iunit) el_codigo
         end if
         return
         end
c
         subroutine transmit_page ( iunit, ans ) ! (Adding unit number)
c ---    <ESC>[p
         character*3     el_codigo
         character*1     ans
         el_codigo = char(27)//char(91)//char(112)
         if ( ans .eq. '*' ) then
            write(iunit,*) el_codigo
            else
            write(iunit) el_codigo
         end if
         return
         end
c
         subroutine transmit_curr_line ( iunit ) ! (Adding unit number)
c ---    <ESC>[1p
         character*4 el_codigo
         el_codigo = char(27)//char(91)//char(49)//char(112)
         write(iunit) el_codigo
         return
         end
c
         subroutine transmit_char_at_cursor ( iunit )    ! (unit number)
c ---    <ESC>[2p
         character*4 el_codigo
         el_codigo=char(27)//char(91)//char(50)//char(112)
         write(iunit) el_codigo
         return
         end
c
         subroutine transmit_25th_line ( iunit ) ! (Adding unit number)
c ---    <ESC>[3p
         character*10 el_codigo
         el_codigo=char(27)//char(91)//char(51)//char(112)
         write(iunit) el_codigo
         return
         end
c
         subroutine cursor_up ( el_codigo, is, ie, n ) ! (May-10)
c ---    <ESC>[PnA
         character*(*) el_codigo
            nsave = n
         icounter = 0
1        icounter = icounter + 1
         if ( nsave .ge. 100 ) then
            nsave = nsave/10
            goto 1
         end if
         icoc = nsave/10
         imod = mod(nsave,10)
         el_codigo(is:is+1) = char(27)//char(91)
         if( icoc .eq. 0 ) then
            el_codigo(is+2:is+3) = char(nsave+48)//char(65)
            ie = is + 4
            return
                        else
```

```
            el_codigo(is+2:is+2) = char(48+icoc)
            el_codigo(is+3:is+4) = char(48+imod)//char(65)
            ie = is + 5
            return
         end if
         end
c
         subroutine cursor_down ( el_codigo, is, ie, n ) ! (May-10)
c ---    <ESC>[PnB
         character*(*) el_codigo
            nsave = n
         icounter = 0
1        icounter = icounter + 1
         if ( nsave .ge. 100 ) then
            nsave = nsave/10
            goto 1
         end if
         icoc = nsave/10
         imod = mod(nsave,10)
         el_codigo(is:is+1) = char(27)//char(91)
         if( icoc .eq. 0 ) then
            el_codigo(is+2:is+3) = char(nsave+48)//char(66)
            ie = is + 4
            return
                           else
            el_codigo(is+2:is+2) = char(48+icoc)
            el_codigo(is+3:is+4) = char(48+imod)//char(66)
            ie = is + 5
            return
         end if
         end
c
         subroutine cursor_right ( el_codigo, is, ie, n ) ! (May-10)
c ---    <ESC>[PnC
         character*(*) el_codigo
            nsave = n
         icounter = 0
1        icounter = icounter + 1
         if ( nsave .ge. 100 ) then
            nsave = nsave/10
            goto 1
         end if
         icoc = nsave/10
         imod = mod(nsave,10)
         el_codigo(is:is+1) = char(27)//char(91)
         if( icoc .eq. 0 ) then
            el_codigo(is+2:is+3) = char(48+nsave)//char(67)
            ie = is + 4
            return
                           else
            el_codigo(is+2:is+2) = char(48+icoc)
            el_codigo(is+3:is+4) = char(48+imod)//char(67)
            ie = is + 5
            return
         end if
         end
c
         subroutine cursor_left ( el_codigo, is, ie, n ) ! (May-10)
c ---    <ESC>[PnD
         character*(*) el_codigo
```

```
            nsave = n
          icounter = 0
1         icounter = icounter + 1
          if ( nsave .ge. 100 ) then
             nsave = nsave/10
             goto 1
          end if
          icoc = nsave/10
          imod = mod(nsave,10)
          el_codigo(is:is+1) = char(27)//char(91)
          if( icoc .eq. 0 ) then
             el_codigo(is+2:is+3) = char(48+nsave)//char(68)
             ie = is + 4
             return
                          else
             el_codigo(is+2:is+2) = char(48+icoc)
             el_codigo(is+3:is+4) = char(48+imod)//char(68)
             ie = is + 5
             return
          end if
          end
c
          subroutine cursor_home( el_codigo, is, ie )
c ---     <ESC>[H
          character*(*) el_codigo
          el_codigo(is:is+2) = char(27)//char(91)//char(72)
          ie = is + 3
          return
          end
c
          subroutine enter_alt_char_set( el_codigo, is, ie )
c ---     <ESC>(1
          character*(*) el_codigo
          el_codigo(is:is+2) = char(27)//char(40)//char(49)
          ie = is + 3
          return
          end
c
          subroutine exit_alt_char_set( el_codigo, is, ie )
c ---     <ESC>(0<ESC>[11m
          character*(*) el_codigo
          el_codigo(is:is+2) = char(27)//char(40)//char(48)
          is = is + 3
          call exit_graph_mode( el_codigo, is, ie )
          return
          end
c
          subroutine enter_alt_char_graph_set( el_codigo, is, ie )
c ---     <ESC>(2
          character*(*) el_codigo
          el_codigo(is:is+2) = char(27)//char(40)//char(50)
          ie = is + 3
          return
          end
c
          subroutine exit_alt_char_graph_set( el_codigo, is, ie )
c ---     <ESC>(0<ESC>[11m
          character*(*) el_codigo
          el_codigo(is:is+2) = char(27)//char(40)//char(48)
          is = is + 3
```

```
      call exit_graph_mode( el_codigo, is, ie )
      return
      end
c
      subroutine cursor_off( iunit )
c ---  <ESC>[>5h
      character*10 el_codigo
      el_codigo(1:5)=char(27)//char(91)//char(62)//char(53)//char(104)
      write(iunit) el_codigo
      return
      end
c
      subroutine cursor_on( iunit )
c ---  <ESC>[>5l
      character*10 el_codigo
      el_codigo(1:5)=char(27)//char(91)//char(62)//char(53)//char(108)
      write(iunit) el_codigo
      return
      end
c
      subroutine save_cursor_position( el_codigo, is, ie )
c ---  <ESC>[s
      character*(*) el_codigo
      el_codigo(is:is+2) = char(27)//char(91)//char(115)
      ie = is + 3
      return
      end
c
      subroutine cursor_to_save_position( el_codigo, is, ie )
c ---  <ESC>[u
      character*(*) el_codigo
      el_codigo(is:is+2) = char(27)//char(91)//char(117)
      ie = is + 3
      return
      end
```

```
c +-------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center        !
c !                        Hunstville, Alabama                        !
c !                              1985                                 !
c +-------------------------------------------------------------------+

c ********************************************************************
c *                                                                  *
c *                     Subroutine INITIALIZE                        *
c *                                                                  *
c ********************************************************************

c       Last revised by Bor-Jau Hsieh (Andy) &
c                       Juan J. Rodriguez-Moscoso (Jun-22-85)

        SUBROUTINE INITIALIZE ( Nrt, ndim, t0, tf, deltat, Y )

c-------------------------------------------------------------------
c*              COMMON AREAS DEFINITIONS AND DIMENSIONS ARRAYS      *
c-------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

c ***   Common Areas Definitions:
        COMMON / FLAG / FLAG(7)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)
        COMMON / STEP / PMO(3)          , RTini(3)      , RTend(3)      ,
     +                  DT(3)           , ST(3)         , PT(3)         ,
     +                  RT(3)           , RTimin(3)     , RTemin(3)     ,
     +                  DTmin(3)        , Thpeak(3)
        COMMON / FRQ1 / Amp             , Omega         , Phase         ,
     +                  ts
        COMMON / FRQ2 / THAMP(3)        , THFEE(3)      , THAOL(3)      ,
     +                  THPOL(3)        , THS(3)        , THC(3)        ,
     +                  X1(3)           , X2(3)         ,
     +                  OMAMP(3)        , OMFEE(3)      , OMAOL(3)      ,
     +                  OMPOL(3)        , OMS(3)        , OMC(3)        ,
     +                  X3(3)           , X4(3)         , XN
        COMMON / FRQ3 / Freq            , Const         , Tau
        COMMON / PI   / PI

        DIMENSION       Y(10)           , Ytemp(10)
        LOGICAL*1       FLAG
        REAL*8          Kp      , Kd    , INMAT , INMATV

c-------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c-------------------------------------------------------------------
        DO I=1, 10                                  ! Store initial states.
           Ytemp(I) = Yin(I)
        END DO

        IF ( FLAG(3) )
           THEN                                     ! IMPULSE Response.
            IF ( Nrt .EQ. 1 ) THEN
               DO i=1, 10                           ! Reset initial states.
                  Y(i) = Ytemp(i)
               END DO
               IF ( FLAG(4) ) THEN                  ! Check Quaternion.
                  ndim = 7
```

```
         CALL INIT_QUATERN ( Y )
      END IF
   END IF
   t0 = (Nrt-1)*(tf+deltat) + (2-Nrt)*t0
   tf = (Nrt-1)*Bound(2) + (2-Nrt)/Bound(7)
   DO k=1, 3
      Thcom(k) = (2-Nrt)*Bound(7)*Bound(k+9)
      Omcom(k) = 0.0d0
   END DO
   RETURN
END IF

Y( 1) = Ytemp( 1)
Y( 2) = Ytemp( 2)
Y( 3) = Ytemp( 3)
Y( 4) = Ytemp( 4)
Y( 5) = Ytemp( 5)
Y( 6) = Ytemp( 6)
Y( 7) = Ytemp( 7)
Y( 8) = Ytemp( 8)
Y( 9) = Ytemp( 9)
Y(10) = Ytemp(10)

IF ( FLAG(4) )                          ! Check Quaternion
   THEN
    ndim = 7
    CALL INIT_QUATERN ( Y )
END IF

IF ( FLAG(1) ) RETURN                   ! STEP Response.

DO I = 1, 3                             ! FREQUENCY Response.
   THC(I) = 0.d0                        ! Clear Temporary
   THS(I) = 0.d0                        ! Storages.
   OMC(I) = 0.d0
   OMS(I) = 0.d0
END DO

  Freq = Freq * Const           ! Compute current frequency.
Period = 1.d0/Freq              ! Compute Period.
deltat = 1.d0/Bound(3)          ! deltat is fixed.
 nstep = INT( Period/deltat )           ! Compute integer # of
                                        ! deltat's in 1 period
Period = DFLOAT( nstep ) * deltat       ! Truncate Period.
 Omega = 2.d0 * PI / Period             ! Compute Angular Freq.

   ts = 6.d0 * Tau                      ! Compute Integrate
                                        !   start time.
   tf = 6.d0 * Tau + Period             ! Compute Integrate
RETURN
END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     Subroutine INIT_QUATERN                        *
c *                                                                    *
c **********************************************************************

c ***   This routine computes the initial quaternion given the initial
c       Roll-Pitch-Yaw angles, or Euler angles, in terms of the V frame.

c ***   Written by Juan J. Rodriguez-Moscoso (May-02-85)
c       Revised on May-12-85
c                on May-13-85    : Saving initial values RPY.
c                on May-18-85    : Cleaning up the routine.
c                on Jun-02-85    : Program restructure.

        SUBROUTINE INIT_QUATERN ( Y )

c--------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                   *
c--------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

c ***   Dimensioning of Arrays
        DIMENSION       QO(4),          ! Initial Quaternion Storage
     +                  Y(10)           ! Vector of states

c--------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c--------------------------------------------------------------------

c ***   The initial RPY angles are stored in Y(7), Y(8), and Y(9) of the
c       Y(i) array. The initial quaternion will be stored in QO(i) where
c       i=1,2,3, and 4. After quaternion's computations, the initial qua-
c       ternion will be stored in Y(i), i=4,5,6,7.

c       Computing Scalar Part of the initial Quaternion: QO(4).
        QO( 4) = DCOS(Y(7)/2.DO)*DCOS(Y(8)/2.DO)*DCOS(Y(9)/2.DO) -
                 DSIN(Y(7)/2.DO)*DSIN(Y(8)/2.DO)*DSIN(Y(9)/2.DO)

c       Computing Vector Part of the initial Quaternion: QO(i) [i=1,2,3].
        QO( 1) = DSIN(Y(7)/2.DO)*DCOS(Y(8)/2.DO)*DCOS(Y(9)/2.DO) +
                 DCOS(Y(7)/2.DO)*DSIN(Y(8)/2.DO)*DSIN(Y(9)/2.DO)
        QO( 2) = DSIN(Y(8)/2.DO)*DCOS(Y(7)/2.DO)*DCOS(Y(9)/2.DO) +
                 DCOS(Y(8)/2.DO)*DSIN(Y(7)/2.DO)*DSIN(Y(9)/2.DO)
        QO( 3) = DSIN(Y(9)/2.DO)*DCOS(Y(7)/2.DO)*DCOS(Y(8)/2.DO) -
                 DCOS(Y(9)/2.DO)*DSIN(Y(7)/2.DO)*DSIN(Y(8)/2.DO)

c ***   Saving initial values of Theta (RPY)
        Y(10) = Y( 9)
        Y( 9) = Y( 8)
        Y( 8) = Y( 7)

c ***   Storing the initial Quaternion into Y(i) [i=4,5,6,7]
        Y( 7) = QO( 4)
```

```
      Y( 6) = QQ( 3)
      Y( 5) = QO( 2)
      Y( 4) = QO( 1)

      RETURN
      END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     SUBROUTINE INTEGRATION                         *
c *                                                                    *
c **********************************************************************

c ***   Written by Juan J. Rodriguez-Moscoso &
c                  Bor-Jau Hsieh (Andy)          (May-02-85)
c       Revised on May-06-85
c                on May-12-85     : Adding aux. Return
c                on Jun-02-85     : Program restructure.

c ***   This routine checks out what type of integration method has been
c       selected by the user. Then, it makes the appropriate calls to
c       the integration routines.

c ***   The type of integration method is indicated by:
c
c                If  FLAG(5) = .TRUE.     then    EULER
c                If  FLAG(6) = .TRUE.     then    RUNGE-KUTTA
c                If  FLAG(7) = .TRUE.     then    PREDICTOR-CORRECTOR

        SUBROUTINE INTEGRATION ( ndim, t0, tf, deltat, Y, Ydot, *, * )

c----------------------------------------------------------------------
c!              COMMON AREAS AND DIMENSION OF ARRAYS                  !
c----------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)
        COMMON / FLAG / FLAG(7)

        DIMENSION       Y(10)           , Ydot(10)
        REAL*8          Kp      , Kd    , INMAT , INMATV
        LOGICAL*1       FLAG

c----------------------------------------------------------------------
c*                      INTEGRATION                                   *
c----------------------------------------------------------------------
c ***   Checking out type of integration to be applied.
c ---   EULER Integration when FLAG(5) = .TRUE.
        IF ( FLAG(5) )
     .     THEN
            CALL EULER( ndim, t0, tf, deltat, Y, Ydot )
            RETURN
        END IF

c ---   RUNGE-KUTTA FOURTH ORDER Integration when FLAG(6) = .TRUE.
        IF ( FLAG(6) )
     .     THEN
            EPS = Bound( 4)
            CALL RUNGE_KUTTA ( ndim, t0, tf, deltat, EPS, Y, Ydot, &100 )
            RETURN
```

```
          END IF

c ---     PREDICTOR-CORRECTOR Integration when FLAG(7) = .TRUE.
          IF ( FLAG(7) )
             THEN
              EPS = Bound( 4)
              CALL PRED_CORREC ( ndim, t0, tf, deltat, EPS, Y, Ydot, &110 )
              RETURN
          END IF

c ---     RETURN codes in case of Integration problems.
100       RETURN 1
110       RETURN 2
          END
```

```
c +-------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center        !
c !                        Hunstville, Alabama                        !
c !                              1985                                 !
c +-------------------------------------------------------------------+

c ********************************************************************
c *                                                                  *
c *     GENeric SIMulation program (FORTRAN-77 portion of NESS)      *
c *                                                                  *
c ********************************************************************

c ***   This program has been designed for solving in a generic form the
c       simulation problem stated by the following set of equations:
c       PROTOTYPE 0:
c       ------------
c               Torque(i) = Kp(i,j)*[ Thcom(i)-Thact(i) ] +
c                           Kd(i,j)*[ Omcom(i)-Omact(i) ]    (ConTRoLLER)
c
c               Ydot(i) = INMATV(i,j)*[ Torque(i) -
c                           Omega(i) X INMAT(i,j)*Omega(i) ] (BOdy
c                                                              DYnamics)
c       with the assumptions:
c               Steering Dynamics : Unity
c               Actuator & Torque : Unity
c                         Sensors : Unity

c       PROTOTYPE 1:
c       ------------
c                  Qdot(i) = F( t, Q, Omega_v ) (Quaternion
c                                                  rate-equations)
c               Theta_v(i) = Q( [VR] )          (Rotation Matrix VR to
c                                                  determine Theta_v)
c       DESCRIPTION OF INPUT VARIABLES:
c       ===============================
c                           Step Response ! Freq. Response
c                           ------------- ! -------------
c               Bound( 1) = t0            . t0           (initial time)
c                    ( 2) = tf            . Tau          (final time & tau)
c                    ( 3) = deltat        . 1/deltat     (increment time)
c                    ( 4) = Epsilon       . Epsilon      (error for Integ)
c                    ( 5) = 0.0           . 0.0          (Stop Condition)
c                    ( 6) = % Steady S.   . ANLZ axis    (Steady State E. &
c                                                         axis for analyze)
c                    ( 7) = blank         . blank
c                    ( 8) = blank         . Amplitude    (Inp. Sinu. Amp.)
c                    ( 9) = balnk         . Omega_lo     (lowest Freq.)
c                    (10) = Omega_x (Inp) . Phase        (Omega_x Input &
c                                                         Phase for FRQ)
c                    (11) = Omega_y   "   . Ndec         (Omega_y Input &
c                                                         (# of decades)
c                    (12) = Omega_z   "   . Nsd          (Omega_z Input &
c                                                         (# samp. f/dec.)
c                    (13) = Theta_x   "   . Theta_x      (Inp. axis = 0,1
c                    (14) = Theta_y   "   . Theta_y      for FRQ, any value
c                    (15) = Theta_z   "   . Theta_z      for STP.)
c
c                           PROTOTYPE 0   ! PROTOTYPE 1
c                           -----------   . -----------
c                     Y( 1) = Omega_x(0)  . Omega_x(0)   (Initial Omega_x)
```

```
c                    ( 2) = Omega_y(0)    . Omega_y(0)  (Initial Omega_y)
c                    ( 3) = Omega_z(0)    . Omega_z(0)  (Initial Omega_z)
c                    ( 4) = Theta_x(0)    . blank       (Initial Theta_x)
c                    ( 5) = Theta_y(0)    . blank       (Initial Theta_y)
c                    ( 6) = Theta_z(0)    . blank       (Initial Theta_z)
c                    ( 7) = blank         . ROLL(0)     (Init. Roll angle)
c                    ( 8) = blank         . YAW(0)      (Init. Yaw angle)
c                    ( 9) = blank         . PITCH(0)    (Init. Pitch ang.)
c                    (10) = blank         . blank

c ***   Written by Bor-Jau Hsieh (Andy) &
c                  Juan J. Rodriguez-Moscoso (01-May-85)

c       Last revised on 31-May-85
c
c ---------------------------------------------------------------------

        PROGRAM GENSIM

c ---------------------------------------------------------------------
c *             COMMON AREAS DEFINITIONS AND ARRAYS DIMENSIONING      *
c ---------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)
        COMMON / FLAG / FLAG(7)
        COMMON / PI   / PI

        REAL*8          Kp      , Kd    , INMAT , INMATV
        LOGICAL*1       FLAG

c ---------------------------------------------------------------------
c *                     COMPUTATIONS                                  *
c ---------------------------------------------------------------------
c ***   Computing PI:
        PI = DATAN2( 0.0D0 , -1.0D0 )

c ***   Reading in the input file: "SIMULA.INP"
        OPEN (unit = 1, status ='old', file = 'SIMULA.INP')

        READ( 1,* )     FLAG    ,       ! Type of response & integration.
     +                  Bound   ,       ! Boundaries of conditions.
     +                  Yin     ,       ! Initial States.
     +                  Kp      ,       ! Kp Matrix.
     +                  Kd      ,       ! Kd Matrix.
     +                  INMAT           ! Inertia Matrix.

c ***   Closing Input Unit
        CLOSE(unit=1)

c ***   Open output files according to specified response type.
        CALL OUTPUT_FILES

c ***   initiate integration routine                    ! May-31-85
        CALL SIMULATION ( ICODE )

c ***   Closing files and deletion of FOR017.DAT & FOR018.DAT.
        CLOSE( unit=17, DISP='DELETE' )                 ! May-29-85
        CALL EXIT
```

```
c ***   Check return codes
        IF ( ICODE ) 10,30,20

c ***   If ICODE < 0  then, No Response Analysis was chosen.
10      CONTINUE
        WRITE(6,15)
15      FORMAT(1HO,'*************** ERROR DETECTED ***************'/1HO,
     +             '------ NO RESPONSE ANALYSIS WAS CHOSEN ------'/1H ,
     +             '  FORTRAN STOP IS FOUND FROM THIS EXECUTION  ')
        STOP

c ***   If ICODE > 0  then, Error detected during the Integration process
c                           or Inversion routine
20      CONTINUE
        GO TO ( 21, 23, 25 ), ICODE
21      WRITE(6,22)
22      FORMAT(1HO,'*************** ERROR DETECTED ***************'/1HO,
     .             '--- THE INERTIA MATRIX IS NOT INVERSIBLE ----'/1H ,
     .             '  FORTRAN STOP IS FOUND FROM THIS EXECUTION.  ')
        STOP
23      WRITE(6,24)
24      FORMAT(1HO,'*************** ERROR DETECTED ***************'/1HO,
     .             '--- NO CONVERGENCE DURING THE INTEGRATION ---'/1H ,
     .             '  PROCESS FOR RUNGE-KUTTA 4TH ORDER SELECTED '/1H ,
     .             '  FORTRAN STOP IS FOUND FROM THIS EXECUTION.  ')
        STOP
25      WRITE(6,26)
26      FORMAT(1HO,'*************** ERROR DETECTED ***************'/1HO,
     .             '--- NO CONVERGENCE DURING THE INTEGRATION ---'/1H ,
     .             '   PROCESS FOR PREDICTOR-CORRECTOR SELECTED  '/1H ,
     .             '  FORTRAN STOP IS FOUND FROM THIS EXECUTION.  ')
        STOP

c ***   If ICODE = 0  then, Execution suceeded.
30      CONTINUE

c ***   End of Simulation.
        END
```

```
c ------------------------------------------------------------------------
c |            Copyright (c) NASA Marshall Space Flight Center           |
c |                        Hunstville, Alabama                           |
c |                              1985                                    |
c ------------------------------------------------------------------------

c ************************************************************************
c *                                                                      *
c *                       SUBROUTINE MATINV                              *
c *                                                                      *
c ************************************************************************

c ***   Subroutine written by Juan J. Rodriguez-Moscoso (18-Apr-85)

      SUBROUTINE MATINV ( A, N, ICODE )

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION       A(N,N), MU(25)

      ICODE = 0
      DO 5 I=1,N
         MU(I) = I
5     CONTINUE

      DO 100 I=1,N
         IP1 = I+1
         IF( IP1 .LE. N ) THEN
              L = I
              AMAX = DABS( A(I,I) )
              DO 30 K=IP1,N
                 IF( AMAX .LT. DABS(A(K,I)) ) THEN
                      L = K
                      AMAX = DABS( A(K,I) )
                 END IF
30            CONTINUE

              IF( L .EQ. I ) GOTO 50
              K = MU(I)
              MU(I) = MU(L)
              MU(L) = K
              DO 40 J=1,N
                 P = A(I,J)
                 A(I,J) = A(L,J)
                 A(L,J) = P
40            CONTINUE
         END IF

50       P = A(I,I)

C ***   Testing Singularity
         IF( DABS(P) .LE. 0.28D-16 ) THEN
              ICODE = 1
              RETURN
         END IF

C ***   Computing the inverse of the A matrix

         DO 60 J=1,N
            A(I,J) = A(I,J)/P
60       CONTINUE
```

```
            DO 80 K=1,N
               IF( K .EQ. I ) GOTO 80
               DO 70 J=1,N
                  IF( J .EQ. I ) GOTO 70
                  A(K,J) = A(K,J)-A(K,I)*A(I,J)
70             CONTINUE
80          CONTINUE

            DO 90 K=1,N
               A(K,I) = -A(K,I)/P
90          CONTINUE

            A(I,I) = -A(I,I)
100      CONTINUE

         DO 140 J=1,N

            DO 110 K=J,N
               IF( MU(K) .EQ. J ) GOTO 120
110         CONTINUE

120         IF( K .NE. J ) THEN
               MU(K) = MU(J)
               DO 130 I=1,N
                  P = A(I,K)
                  A(I,K) = A(I,J)
                  A(I,J) = P
130            CONTINUE
            END IF
140      CONTINUE

         RETURN
         END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     Subroutine NEW_VALUES                          *
c *                                                                    *
c **********************************************************************

c ***   This subroutine performs the following operations:
c               1.- Normalization of the Quaternion by applying the
c                   following relations:
c                       N = DSQRT*( Q1**2+Q2**2+Q3**2+Q4**2 )
c                   and,
c                       Q = Qi/N
c               2.- Computation of the [VR] Matrix from the updated
c                   quaternion.
c               3.- Computation of the new set of angular positions from
c                   the [VR] matrix by applying eq. 2.9 (Attitude Module
c                   representation paper sent by K. Fernandez).

c ***   Written by Juan J. Rodriguez-Moscoso ( 04-May-85 )
c       Last revised on 02-Jun-85

        SUBROUTINE NEW_VALUES ( ndim, Y )

c--------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                   *
c--------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON / FLAG / FLAG(7)

        DIMENSION       Y(10),          ! Vector of states.
     +                  VR(3,3)         ! The [VR] matrix

        LOGICAL*1       FLAG

c--------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c--------------------------------------------------------------------
c ***   Checking out for Quaternion flag
        IF( .NOT. FLAG(4) ) RETURN

c ***   Computing the quaternion norm and update of Quaternion.
        the_norm = Y( 4)*Y( 4) + Y( 5)*Y( 5) + Y( 6)*Y( 6) + Y( 7)*Y( 7)
        the_norm = DSQRT( the_norm )
           Y( 4) = Y( 4)/the_norm
           Y( 5) = Y( 5)/the_norm
           Y( 6) = Y( 6)/the_norm
           Y( 7) = Y( 7)/the_norm

c ***   Computing the [VR] matrix
        DO i=1,3                                    ! (May-10)
           DO j=1,3                                 ! (May-10)
              VR(i,j) = Y(i+3)*Y(j+3)               ! (May-10)
           END DO
        END DO
```

```
      DO i=1,3                                  ! (May-10)
         DO j=1,3                               ! (May-10)
                  k = j*( -66+i*(98-27*i)+j*(13+i*(-21+6*i)) ) +
     +                i*( -102+27*i ) + 76
               kk = JIABS( k )
            ksign = JISIGN(1,k)
         VR(i,j) = VR(i,j) + DFLOTJ(ksign)*Y(7)*Y(kk+3)
         VR(i,j) = 2.d0*VR(i,j)
         END DO
      END DO

      DO i=1,3                                  ! (May-10)
         VR(i,i) = VR(i,i) - 1.0d0
      END DO

c ***   The set of values for Theta are extracted from the [VR] matrix.
c       These values will be stored at Y(i) for i=8,9,10.

       temp = DSQRT( VR(2,2)*VR(2,2)+VR(3,2)*VR(3,2) )
      Y( 8) = DATAN2( - VR(3,2), VR(2,2) )
      Y( 9) = DATAN2( -VR(1,3), VR(1,1) )
      Y(10) = DATAN2( VR(1,2), temp )

      RETURN
      END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *              SUBROUTINE OUTPUT_FILES                               *
c *                                                                    *
c **********************************************************************
c
c ***   Written by Juan J. Rodriguez-Moscoso ( 02-May-85 )
c       Last revised by J. Rodriguez-Moscoso &
c                       Bor-Jau Hsieh (Andy) on 02-Jun-85

      SUBROUTINE OUTPUT_FILES

c ---------------------------------------------------------------------
c *             COMMON AREAD DEFINITIONS AND ARRAYS DIMENSIONING       *
c ---------------------------------------------------------------------
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

c ***   Common Areas Definitions:
      COMMON / FLAG / FLAG(7)

c ***   Variables Definitions
      CHARACTER*13    FILEOUT
      CHARACTER*10    EXTO
      CHARACTER*12    EXT1
      CHARACTER*9     EXT2
      CHARACTER*5     THETA, OMEGA, NUMER
      CHARACTER*3     CHA, PLT, AMPLIT, PHAS, FRQ, IMP, STP, OUT
      CHARACTER*1     ZERO, DOT
      LOGICAL*1       FLAG


c ***   Initializing a few variables
      DATA            THETA/'THETA'/, OMEGA/'OMEGA'/, NUMER/'NUMER'/,
     +                PLT/'PLT'/, AMPLIT/'AMP'/, PHAS/'PHA'/,
     +                FRQ/'FRQ'/, IMP/'IMP'/, STP/'STP'/, OUT/'OUT'/,
     +                CHA/'CHA'/, ZERO/'0'/, DOT/'.'/

c ---------------------------------------------------------------------
c *                     COMPUTATIONS                                   *
c ---------------------------------------------------------------------
      EXTO = THETA//OMEGA
      EXT1 = CHA//PLT//AMPLIT//PHAS
      EXT2 = STP//FRQ//IMP

c ***   Open Output Files of Simulation.
c       The output files are given by the following relationships:
c       NUMERICAL OUTPUT:
c               -> UNIT = 1     ; NUMEROOUT.STP (Step Response)
c               ->      = 2     ; NUMEROOUT.FRQ (Freq Response)
c               ->      = 3     ; NUMEROOUT.IMP (Impulse Response)
c       STEP RESPONSE:
c               -> UNIT = 7     ; THETAOCHA.STP (Characteristics)
c               ->      = 8     ; THETAOPLT.STP (Plot of Theta)
c               ->      = 9     ; OMEGAOPLT.STP (Plot of Omega)
```

```
c       FREQ. RESPONSE: (BODE plotting)
c               -> UNIT = 10    ; THETAOAMP.FRQ (Amplitude of Theta)
c               ->      = 11    ; THETAOPHA.FRQ (Phase of Theta)
c               ->      = 12    ; OMEGAOAMP.FRQ (Amplitude of Omega)
c               ->      = 13    ; OMEGAOPHA.FRQ (Phase of Theta)
c       IMPULSE RESPONSE:
c               -> UNIT = 14    ; THETAOCHA.IMP (Characteristics)
c               ->      = 15    ; THETAOPLT.IMP (Plot of Theta)
c               ->      = 16    ; OMEGAOPLT.IMP (Plot of Omega)
c       TEMPORARY STORAGE:
c               ->      = 17    ; FOR017.DAT    (Time series data)
c       PRINTER OUTPUT:
c               ->      = 18    ; PRINTER.DAT   (Printer Output)

        DO i = 1, 3
           FILEOUT(1:10) = NUMER//ZERO//OUT//DOT
           IF ( FLAG(i) ) THEN
                i1 = 3*i - 2
                i2 = i1 + 2
              iunit = i*(3+i)/2 + 4             ! (May-04-85)
              FILEOUT(11:13) = EXT2(i1:i2)
              OPEN( unit=i, status='new', name=FILEOUT )

              k = i*( 4-i )                     ! (May-04-85)
              DO j=1,k
                 j0 = j*( -155+j*(75-10*j) )/6 +16     ! (May-04-85)
                 j00 = j0 + 4
                 j1 = (i-2)*(i-2)*( j*(15-3*j)/2-5 ) + ! (May-04-85)
     +                (i-1)*(3-i)*( j*( 34+j*(-15+j*2) )-14 )
                 j11 = j1 + 2
                 FILEOUT(1:5) = EXT0(j0:j00)
                 FILEOUT(7:9) = EXT1(J1:J11)
                 iunit = iunit + 1
                 OPEN( unit=iunit, status='new', name=FILEOUT )
              END DO
           END IF
        END DO

        OPEN( unit=17, status='new' )
        OPEN( unit=18, status='new', name='PRINTER.DAT' )

        RETURN
        END
```

```
c +--------------------------------------------------------------------+
c |            Copyright (c) NASA Marshall Space Flight Center         |
c |                        Hunstville, Alabama                         |
c |                              1985                                  |
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     Subroutine OUTPUT_17                           *
c *                                                                    *
c **********************************************************************

c ***   This subroutine handles all related to output of Simulation.  It
c       is recording partial results.
c
c       Written by Juan J. Rodriguez-Moscoso ( 22-Jan-85 )
c       Last revised on 13-May-85

        SUBROUTINE OUTPUT_17 ( ndim, t, deltat, Y, Ydot )

c--------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSIONS ARRAYS                   *
c--------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON / FLAG / FLAG

        DIMENSION       Y(10)   , Ydot(10)
        LOGICAL*1       FLAG(7)

c--------------------------------------------------------------------
c*                      COMPUTATIONS                                 *
c--------------------------------------------------------------------
c ***   No temporary data generated for FREQUENCY response.
        IF ( FLAG(2) ) RETURN

c ***   Checking out if "t" is = to "t + m*deltat"
c       RES = DMOD( t, deltat )
c       IF ( RES .GT. 0.1D-12 )
c     .    THEN
c            IF ( DABS( RES - deltat ) .GT. 0.1D-12 ) RETURN
c       END IF

c ***   Checking FLAG(4) for Quaternion
        IF ( FLAG(4) )
     .    THEN
            WRITE(17,*) t               , ! current time,
     +                  ( Y(j), j=1,10 ) ! Omega and Theta
            RETURN
           ELSE
            WRITE(17,*) t, ( Y(j), j=1,ndim )
            RETURN
        END IF

        END
```

```
c +-------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center        !
c !                        Hunstville, Alabama                        !
c !                              1985                                 !
c +-------------------------------------------------------------------+

c ********************************************************************
c *                                                                  *
c *                     SUBROUTINE PRED_CORREC                       *
c *                                                                  *
c ********************************************************************

c ***   Subroutine to solve a system of first order ordinary general dif-
c       ferential equations with given initial values.

c ***   This method is based on Hammings modified Predictor-Corrector me-
c       thod. This is a fourth order method using 4 preceeding points for
c       computation of a new vector Y of the dependent variables.
c       The adjustment of the initial increment and computation of the s-
c       tarting values is done by a Fourth-order Runge-Kutta method  sug-
c       gested by Ralston.

c ***   Written by Juan J. Rodriguez-Moscoso on 21-Apr-85.
c       Last revised on 01-Jun-85

        SUBROUTINE PRED_CORREC ( ndim, t0, tf, tincr, EPS, Y, Ydot, * )

c-------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION OF ARRAYS                *
c-------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DIMENSION       Y(10)           , Ydot(10)      , aux(16,10)

c-------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c-------------------------------------------------------------------
           n = 1
        ihlf = 0
           t = t0
        tend = tf
      deltat = tincr
       ICODE = 0

        DO I=1,ndim
           aux(16,i) = 0.0D0
           aux(15,i) = Ydot(i)
            aux(1,i) = Y(i)
        END DO

        IF( deltat*( tend - t ) ) 30, 20, 40
20      ihlf = 12
        GO TO 40
30      ihlf = 13
40      CONTINUE
        CALL SYS_DYNAM ( ndim, t, Y, Ydot )
        CALL OUTPUT_17 ( ndim, t, tincr, Y, Ydot )
        CALL CHECK_ERROR ( ihlf, ICODE )
        IF ( ICODE .NE. 0 ) RETURN 1
        IF ( ihlf .GT. 0 ) RETURN
```

```
      DO i=1,ndim
         aux(8,i) = Ydot(i)
      END DO

      isw = 1
      GOTO 1000

90    t = t + deltat
      DO i=1,ndim
         aux(2,i) = Y(i)
      END DO
110   ihlf = ihlf + 1
         t = t - deltat
      DO i=1,ndim
         aux(4,i) = aux(2,i)
      END DO
      deltat = 0.5d0*deltat
           n = 1
         isw = 2
      GOTO 1000

130   t = t + deltat
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )
      n = 2
      DO i=1,ndim
         aux(2,i) = Y(i)
         aux(9,i) = Ydot(i)
      END DO
      isw = 3
      GOTO 1000

150   DELT = 0.0D0
      DO i=1,ndim
         DELT = DELT + aux(15,i)*DABS( Y(i)-aux(4,i) )
       END DO
      DELT = 0.06666666666666667*DELT
      IF ( DELT - EPS ) 190, 190, 170
170   IF ( ihlf - 10 ) 110, 180, 180
180   ihlf = 11
         t = t + deltat
      GO TO 40

190   t = t + deltat
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )
      DO i=1,ndim
          aux(3,i) = Y(i)
         aux(10,i) = Ydot(i)
      END DO
        n = 3
      isw = 4
      GOTO 1000

210   n = 1
      t = t + deltat
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )
      t = t0
      DO i=1,ndim
         aux(11,i) = Ydot(i)
              Y(i) = aux(1,i) + deltat*( 0.375d0*aux(8,i) +
     +                0.7916666666666667*aux(9,i) -
```

```
     +           0.20833333333333333*aux(10,i) +
     +           0.04166666666666667*Ydot(i) )
         END DO

230      t = t + deltat
         n = n + 1
         CALL SYS_DYNAM ( ndim, t, Y, Ydot )
         CALL OUTPUT_17 ( ndim, t, tincr, Y, Ydot )
         CALL CHECK_ERROR ( ihlf, ICODE )
         IF ( ICODE .NE. 0 ) RETURN 1
         IF ( n .GE. 4 ) GOTO 2000
         DO i=1,ndim
            aux(n,i) = Y(i)
           aux(n+7,i) = Ydot(i)
         END DO

         IF ( n - 3 ) 270, 290, 2000
270      DO i=1,ndim
           DELT = 4.0d0*aux(9,i)
           Y(i) = aux(1,i) + deltat*( aux(8,i)+DELT+aux(10,i) )/3.d0
         END DO
         GO TO 230

290      DO i=1,ndim
            DELT = aux(9,i) + aux(10,i)
            DELT = DELT + DELT + DELT
            Y(i) = aux(1,i)+0.375d0*deltat*( aux(8,i)+DELT+aux(11,i) )
         END DO
         GO TO 230

c ***    The following part of the routine computes by means of Runge-
c        Kutta method the starting values for the not self-starting
c        Predictor-corrector method.

1000     CONTINUE
         DO i=1,ndim
                  z = deltat*aux(n+7,i)
           aux(5,i) = z
               Y(i) = aux(n,i) + 0.4d0*z
         END DO
         z = t + 0.4d0*deltat
         CALL SYS_DYNAM ( ndim, z, Y, Ydot )

         DO i=1,ndim
                  z = deltat*Ydot(i)
           aux(6,i) = z
               Y(i) = aux(n,i)+0.29697760924775360d0*aux(5,i)+
     +                           0.15875964497103583d0*z
         END DO
         z = t + 0.45573725421878943d0*deltat
         CALL SYS_DYNAM ( ndim, z, Y, Ydot )

         DO i=1,ndim
                  z = deltat*Ydot(i)
           aux(7,i) = z
               Y(i) = aux(n,i)+0.21810038822592047d0*aux(5,i)-
     +                           3.05096514869293080d0*aux(6,i)+
     +                           3.83286476046701030d0*z
         END DO
         z = z + deltat
```

```
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )

      DO i=1,ndim
         Y(i) = aux(n,i)+0.17476028226269037d0*aux(5,i)-
     +                   0.55148066287873294d0*aux(6,i)+
     +                   1.20553559939652350d0*aux(7,i)+
     +                   0.17118478121951903d0*deltat*Ydot(i)
      END DO
      GO TO ( 90, 130, 150, 210 ), isw

2000  istep = 3
2010  IF ( n - 8 ) 2040, 2020, 2040
2020  DO n=2,7
         DO i=1,ndim
            aux(n-1,i) = aux(n,i)
            aux(n+6,i) = aux(n+7,i)
         END DO
      END DO
      n = 7

2040  n = n + 1
      DO i=1,ndim
         aux(n-1,i) = Y(i)
         aux(n+6,i) = Ydot(i)
      END DO
          t = t + deltat
2060  istep = istep + 1

      DO i=1,ndim
         DELT = aux(n-4,i) + 1.3333333333333333*deltat*( aux(n+6,i) +
     +           aux(n+6,i) - aux(n+5,i) + aux(n+4,i) + aux(n+4,i) )
         Y(i) = DELT - 0.92561983471074380d0*aux(16,i)
         aux(16,i) = DELT
      END DO
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )

c ***  Derivative of Modified Predictor is generated in Ydot.
      DO i=1,ndim
         DELT = 0.125d0*( 9.d0*aux(n-1,i)-aux(n-3,i)+
     +           3.d0*deltat*( Ydot(i)+2.d0*aux(n+6,i)-aux(n+5,i) ) )
         aux(16,i) = aux(16,i) - DELT
         Y(i) = DELT + 0.07438016528925620d0*aux(16,i)
      END DO

c ***  Testing whether "deltat" must be halved or doubled.
      DELT = 0.0D0
      DO i=1,ndim
         DELT = DELT + aux(15,i)*DABS( aux(16,i) )
      END DO
      IF ( DELT - EPS ) 2100, 2220, 2220

c ***  "deltat" cannot be halved, which means that Y(i) are good.
2100  CONTINUE
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )
      CALL OUTPUT_17 ( ndim, t, tincr, Y, Ydot )
      CALL CHECK_ERROR ( ihlf, ICODE )
      IF ( ICODE .NE. 0 ) RETURN 1
      IF ( ihlf .GE. 11 ) RETURN
      IF ( deltat*( t - tend ) .GE. 0.0d0 ) RETURN
      IF ( DABS( t - tend ) .LT. 0.1d0*DABS( t ) ) RETURN
```

```
      IF ( DELT .GT. 0.02d0*EPS ) GOTO 2010

c ***  "deltat" might be doubled if all necessary values are available.
      IF (  ihlf .LE. 0 ) GOTO 2010
      IF (     n .LT. 7 ) GOTO 2010
      IF ( istep .LT. 4 ) GOTO 2010

      imod = istep/2
      IF ( istep .NE. imod + imod ) GOTO 2010
      deltat = deltat + deltat
        ihlf = ihlf - 1
       istep = 0
      DO i=1,ndim
         aux(n-1,i) = aux(n-2,i)
         aux(n-2,i) = aux(n-4,i)
         aux(n-3,i) = aux(n-6,i)
         aux(n+6,i) = aux(n+5,i)
         aux(n+5,i) = aux(n+3,i)
         aux(n+4,i) = aux(n+1,i)
               DELT = aux(n+6,i) + aux(n+5,i)
               DELT = DELT + DELT + DELT
          aux(16,i) = 8.962962962962963d0*( Y(i)-aux(n-3,i) ) -
     +                3.3611111111111111d0*( Ydot(i)+DELT+aux(n+4,i) )*
     +                deltat
      END DO
      GOTO 2010

c ***  "deltat" has to be halved.
2220  ihlf = ihlf + 1
      IF ( ihlf .GT. 10 ) GOTO 2100
      deltat = 0.5d0*deltat
       istep = 0
      DO i=1,ndim
         Y(i) = 0.390625d-2*( 8.d1*aux(n-1,i)+135.d0*aux(n-2,i)+4.d1*
     +           aux(n-3,i)+aux(n-4,i) ) - 0.1171875d0*( aux(n+6,i)-
     +           6.d0*aux(n+5,i)-aux(n+4,i) )*deltat
         aux(n-4,i) = 0.390625d-2*( 12.d0*aux(n-1,i)+135.d0*aux(n-2,i)
     +                +108.d0*aux(n-3,i)+aux(n-4,i) ) - 0.0234375d0*(
     +                aux(n+6,i)+18.d0*aux(n+5,i)-9.d0*aux(n+4,i) )*
     +                deltat
         aux(n-3,i) = aux(n-2,i)
         aux(n+4,i) = aux(n+5,i)
      END DO
         t = t - deltat
      DELT = t - ( deltat + deltat )
      CALL SYS_DYNAM ( ndim, DELT, Y, Ydot )

      DO i=1,ndim
         aux(n-2,i) = Y(i)
         aux(n+5,i) = Ydot(i)
               Y(i) = aux(n-4,i)
      END DO
      DELT = DELT - 2.0d0*deltat
      CALL SYS_DYNAM ( ndim, DELT, Y, Ydot )

      DO i=1,ndim
         DELT = 3.D0*( aux(n+5,i)+aux(n+4,i) )
         aux(16,i) = 8.962962962962963d0*( aux(n-1,i)-Y(i) ) -
     +               3.3611111111111111d0*( aux(n+6,i)+DELT+Ydot(i) )*
     +               deltat
```

```
      aux(n+3,i) = Ydot(i)
END DO
GOTO 2060

END
```

```
c +-------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center        !
c !                        Hunstville, Alabama                        !
c !                              1985                                 !
c +-------------------------------------------------------------------+

c ********************************************************************
c *                                                                  *
c *                     SUBROUTINE POLFRCAL                          *
c *              ( Pseudo Open Loop Frequency Response CALculations) *
c *                                                                  *
c ********************************************************************

c       Last revised by Bor-Jau Hsieh (Andy) &
c                       Juan J. Rodriguez-Moscoso on 02-Jun-85

        SUBROUTINE POLFRCAL( deltat )

c-------------------------------------------------------------------
c*               COMMON AREAS AND DIMENSION ARRAYS                  *
c-------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        COMMON / FLAG / FLAG(7)
        COMMON / USER / user_choice
        COMMON / FRQ1 / Amp             , Omega         , Phase         ,
     +                  ts
        COMMON / FRQ2 / THAMP(3)        , THFEE(3)      , THAOL(3)      ,
     +                  THPOL(3)        , THS(3)        , THC(3)        ,
     +                  X1(3)           , X2(3)         ,
     +                  OMAMP(3)        , OMFEE(3)      , OMAOL(3)      ,
     +                  OMPOL(3)        , OMS(3)        , OMC(3)        ,
     +                  X3(3)           , X4(3)         , XN
        COMMON / PI   / PI

        INTEGER         user_choice
        LOGICAL*1       FLAG

c-------------------------------------------------------------------
c*                       COMPUTATIONS                               *
c-------------------------------------------------------------------
        IF( .NOT. FLAG(2) ) RETURN

        XN = 2.d0*PI/( Omega*deltat )

c ***   Calculate closed-loop amplitude & phase
        DO 10 I=1,3
             THS(I) = THS(I)/XN
             THC(I) = THC(I)/XN
           THAMP(I) = 0.0D0
           THFEE(I) = 0.0D0
           IF ( THS(I) .EQ. 0.0D0 .AND. THC(I) .EQ. 0.0D0 ) GOTO 5
           THAMP(I) = 2.*DSQRT(THS(I)*THS(I)+THC(I)*THC(I))
           THFEE(I) = DATAN2( THC(I), THS(I) )

c ***   Pseudo Open-Loop Computations
           THAOL(I) = THAMP(I)/DSQRT( THAMP(I)*THAMP(I)
     +                + 2.0D0*THAMP(I)*DCOS(THFEE(I)) + 1.0D0 )
           THAOL(I) = 20.0D0*LOG10( THAOL(I) )
              X1(I) = THAMP(I)*DSIN( THFEE(I) )
```

```
              X2(I) = THAMP(I)*DCOS( THFEE(I) ) + 1.0D0
           THPOL(I) = THFEE(I) - DATAN2( X1(I),X2(I) )
           THPOL(I) = THPOL(I)*180.D0/PI - 180.0d0

5          CONTINUE
             OMS(I) = OMS(I)/XN
             OMC(I) = OMC(I)/XN
           OMAMP(I) = 0.0D0
           OMFEE(I) = 0.0D0
           IF ( OMS(I) .EQ. 0.0D0 .AND. OMC(I) .EQ. 0.0D0 ) GOTO 10
           OMAMP(I) = 2.*DSQRT(OMS(I)*OMS(I)+OMC(I)*OMC(I))
           OMFEE(I) = DATAN2(OMC(I),OMS(I))

c ***   Pseudo Open-Loop Computations
           OMAOL(I) = OMAMP(I)/DSQRT( OMAMP(I)*OMAMP(I)
     +                + 2.*OMAMP(I)*DCOS(OMFEE(I)) + 1.0D0 )
           OMAOL(I) = 20.D0*LOG10( OMAOL(I) )
              X3(I) = OMAMP(I)*DSIN(OMFEE(I))
              X4(I) = OMAMP(I)*DCOS(OMFEE(I)) + 1.0D0
           OMPOL(I) = OMFEE(I) - DATAN2( X3(I),X4(I) )
           OMPOL(I) = OMPOL(I)*180.0D0/PI - 180.0d0

10      CONTINUE

        IF ( user_choice .EQ. 1 ) THEN
           WRITE(6,1000) Omega/(2.*PI), Omega, deltat,
     +                 THAOL, THPOL, OMAOL, OMPOL
        END IF

        WRITE (17, *) Omega,
     +             ( THAOL(J), THPOL(J),
     +               OMAOL(J), OMPOL(J), J=1,3 )

1000    FORMAT(1H0,4X,'>~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~',
     +         '~~~~~~~~~~~~~~~~~~~~~~<'/1H ,
     +         5X,'** Current frequency = ',G12.5,' Hz (',G12.5,' r',
     +         'ad/sec );'/1H ,8X,
     +               '   Time increment = ',G12.5,' (deltaT)'/1H ,10X,
     +         'ANGULAR POSITION:'/1H ,10X,16('-')/1H ,18X,'Amplitude ',
     +         '(db)',19X,'Phase (degrees)'/1H ,10X,'  X-axis  ',' Y-',
     +         'axis  ','  Z-axis  ',2X,'  X-axis  ','  Y-axis  ','  Z',
     +         '-axis  '/1H ,10X,3(G10.3),2X,3(G10.3)/1H0,10X,
     +         'ANGULAR RATE:'/1H ,10X,12('-')/1H ,18X,'Amplitude (db)',
     +         19X,'Phase (degrees)'/1H ,10X,'  X-axis  ','  Y-axis  ',
     +         '  Z-axis  ',2X,'  X-axis  ','  Y-axis  ','  Z-axis  '/
     +         1H ,10X,3(G10.3),2X,3(G10.3) )

        END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                      SUBROUTINE QUATERNION                         *
c *                                                                    *
c **********************************************************************

c ***   This subroutine takes care of QUATERNION's computations. The cal-
c       culations present in this routine involve the computation of the
c       Quaternion-Rate equations in terms of the angular rates in V and
c       R frames. In this 1st phase we consider Yr(i) = 0.0.

c ***   Subroutine written by Juan J. Rodriguez-Moscoso on 22-Apr-85
c       Last revised on 10-May-85

        SUBROUTINE QUATERNION ( ndim, Y, Ydot )

c---------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                     *
c---------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON / FLAG / FLAG(7)

        DIMENSION       Y(10),          ! Vehicle Theta & Omega
     +                  Ydot(10),       ! Vehicle Theta & Omega Rates
     +                  Yr(10)          ! Moveable Reference

        LOGICAL*1       FLAG

c---------------------------------------------------------------------
c*                        COMPUTATIONS                                *
c---------------------------------------------------------------------
c ***   Equating explicitly Yr(i) = 0.0
        Yr( 1) = 0.0D0
        Yr( 2) = 0.0D0
        Yr( 3) = 0.0D0
        Yr( 4) = 0.0D0
        Yr( 5) = 0.0D0
        Yr( 6) = 0.0D0
        Yr( 7) = 0.0D0
        Yr( 8) = 0.0D0
        Yr( 9) = 0.0D0
        Yr(10) = 0.0D0

c ***   If FLAG(4) = .FALSE. then no Quaternion computation is performed.
        IF( .NOT. FLAG(4) ) RETURN

c ***   Quaternion's computations are done in arrays Y(i) and Ydot(i),
c       previously to Integration.
c       Thus, we have that:
c               Qdot(1) = Ydot(4) = 0.5*[ Q(4)*( Yv(1)-Yr(1) )
c                                       - Q(3)*( Yv(2)+Yr(2) )
c                                       + Q(2)*( Yv(3)+Yr(3) ) ]
c             . Qdot(2) = Ydot(5) = 0.5*[ Q(3)*( Yv(1)+Yr(1) )
c                                       + Q(4)*( Yv(2)-Yr(2) )
```

```
c                                        - Q(1)*( Yv(3)+Yr(3) ) ]
c              Qdot(3) = Ydot(6) = 0.5*[ - Q(2)*( Yv(1)+Yr(1) )
c                                        + Q(1)*( Yv(2)+Yr(2) )
c                                        + Q(4)*( Yv(3)-Yr(3) ) ]
c              Qdot(4) = Ydot(7) = 0.5*[ - Q(1)*( Yv(1)-Yr(1) )
c                                        - Q(2)*( Yv(2)-Yr(2) )
c                                        - Q(3)*( Yv(3)-Yr(3) ) ]
c ***   Forming the Quaternion Rate equations.

        Ydot(4) = 0.5d0*( Y(7)*( Y(1)-Yr(1) ) - Y(6)*( Y(2)+Yr(2) ) +
     +            Y(5)*( Y(3)+Yr(3) ) )
        Ydot(5) = 0.5d0*( Y(6)*( Y(1)+Yr(1) ) + Y(7)*( Y(2)-Yr(2) ) -
     +            Y(4)*( Y(3)+Yr(3) ) )
        Ydot(6) = 0.5d0*( -Y(5)*( Y(1)+Yr(1) ) + Y(4)*( Y(2)+Yr(2) ) +
     +            Y(7)*( Y(3)-Yr(3) ) )
        Ydot(7) = 0.5d0*( -Y(4)*( Y(1)-Yr(1) ) - Y(5)*( Y(2)-Yr(2) ) -
     +            Y(6)*( Y(3)-Yr(3) ) )

        RETURN
        END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                      Subroutine RESULTS                            *
c *                                                                    *
c **********************************************************************

c ***   This routine takes the data generated by the OUTPUT routine from
c       the temporary files (unit=17) and builds the output files for
c       each type of response.

c ***   Written by Bor-Jau Hsieh (Andy) and
c                  Juan J. Rodriguez-Moscoso on 12-May-85
c       Last revised on 02-Jun-85

        SUBROUTINE RESULTS ( Y )

c---------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION OF ARRAYS                  *
c---------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        COMMON / FLAG / FLAG(7)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)

        COMMON / STEP / PMO(3)          , RTini(3)      , RTend(3)      ,
     +                  DT(3)           , ST(3)         , PT(3)         ,
     +                  RT(3)           , RTimin(3)     , RTemin(3)     ,
     +                  DTmin(3)        , Thpeak(3)
        COMMON / FRQ2 / THAMP(3)        , THFEE(3)      , THAOL(3)      ,
     +                  THPOL(3)        , THS(3)        , THC(3)        ,
     +                  X1(3)           , X2(3)         ,
     +                  OMAMP(3)        , OMFEE(3)      , OMAOL(3)      ,
     +                  OMPOL(3)        , OMS(3)        , OMC(3)        ,
     +                  X3(3)           , X4(3)         , XN
        COMMON / IMPU / ST_imp(3)       , PT_imp(3)     , Thpeak_imp(3)
        COMMON / PI   / PI
        COMMON / SCRP / record_counter  , ndum1         , ndum2

        DIMENSION       Y(10)

        REAL*8          Kp      , Kd    , INMAT , INMATV
        INTEGER         record_counter  , page_counter  , pro_type
        LOGICAL*1       FLAG
        CHARACTER*5     HOLD
        CHARACTER*7     CLS

c---------------------------------------------------------------------
c*                      COMPUTATIONS                                  *
c---------------------------------------------------------------------
        CLS = char(27)//char(91)//char(72)//char(27)//char(91)//char(50)
              //char(74)
        pro_type = 0
```

```
      IF ( FLAG(4) ) THEN
         pro_type = 1
      END IF
      IF ( FLAG(1) )                          ! STEP Response.
         THEN
          DO J = 1, 3
             IF ( Thcom(J) .NE. 0.0D0 ) THEN
                PMO(J) = ( DABS(Thpeak(J)-Thcom(J))/Thcom(J) )*100.0D0
                 RT(J) = RTend(J) - RTini(J)
             END IF
          END DO

          NUNIT1 = 1
          NUNIT2 = 7
          ASSIGN 1000 TO NLINE1
          ASSIGN 1010 TO NLINE2A
          ASSIGN 1020 TO NLINE2B
          ASSIGN 1030 TO NLINE2C
          ASSIGN 1040 TO NLINE3
      END IF

      IF ( FLAG(2) )                  ! FREQUENCY Response.
         THEN
           ndum1 = INT( Bound(11) )
           ndum2 = INT( Bound(12) )
          NUNIT1 = 2
          ASSIGN 2000 TO NLINE1
          ASSIGN 2010 TO NLINE3
      END IF

      IF ( FLAG(3) )                  ! IMPULSE Response.
         THEN
          NUNIT1 = 3
          NUNIT2 = 14
          ASSIGN 3000 TO NLINE1
          ASSIGN 3010 TO NLINE2A
          ASSIGN 3020 TO NLINE3
      END IF

C ***   The next portion of the routine will take care of storing results
C       of simulation.
      REWIND 17
      record_counter = 0      ! Initializing number of records.
        page_counter = 0      ! Initializing number of pages.

C ***   Telling the user to use the NO-SCROLL key to see page by
C       page the output.
      WRITE(NUNIT1,*) CLS
      CALL ENTER_HOLD_SCREEN_MODE ( HOLD, 1, IDUMMY )
      WRITE(NUNIT1,*) HOLD
      WRITE(NUNIT1,900)

C ***   Storing values in Numerical Output
10    CONTINUE
         page_counter = page_counter + 1

         WRITE(NUNIT1,*) CLS
         WRITE(NUNIT1,NLINE1) page_counter

         i = 0
```

```
15        i = i + 1                              ! 14 lines/record
          record_counter = record_counter + 1
          IF ( FLAG(2) )
             THEN
              READ(17,*,END=30) FR, TEMP1, TEMP2,
                                ( Y(j), j=1,10 )
              WRITE(NUNIT1,940) FR, TEMP1, TEMP2,
                                ( Y(j), j=1,10 )
              i = i + 5
              GO TO 20
          END IF

          IF ( FLAG(4) )
             THEN
              READ(17,*,END=30) t                ,      ! Reading All
     +                          ( Y(j), j=1,10 )       ! States
              WRITE(NUNIT1,920) t                ,
     +                          ( Y(j), j=1,3 )  ,      ! Writing All
     +                          ( Y(k), k=8,10 )       ! States
             ELSE
              READ(17,*,END=30) t, ( Y(j), j=1,6 )
              WRITE(NUNIT1,920) t, ( Y(j), j=1,6 )
          END IF

20        IF ( i .LT. 14 ) GO TO 15
          WRITE(NUNIT1,910)
          WRITE(NUNIT1,*) HOLD
        GO TO 10                          ! Return and write new page

30      CONTINUE
c ***   Exiting from hold-screen mode.
        CALL EXIT_HOLD_SCREEN_MODE( HOLD, 1, IDUMMY )
        WRITE(NUNIT1,*) HOLD
        IF ( FLAG(2) ) GO TO 40
        IF ( FLAG(3) ) GO TO 35

c ***   Entering hold-screen mode in unit NUNIT2=7
        WRITE(NUNIT2,*) CLS
        CALL ENTER_HOLD_SCREEN_MODE( HOLD, 1, IDUMMY )
        WRITE(NUNIT2,*) HOLD
        WRITE(NUNIT2,900)

c ***   Storing Characteristics of Step Response in THETAOCHA.STP
        WRITE(NUNIT2,*) CLS
        WRITE(NUNIT2,*) HOLD
        WRITE(NUNIT2,NLINE2A) ( PMO(i), i=1,3 ),
     +                        ( PT(i),  i=1,3 )
        WRITE(NUNIT2,*) CLS
        WRITE(NUNIT2,*) HOLD
        WRITE(NUNIT2,NLINE2B) ( RT(i) , i=1,3 ),
     +                        ( DT(i) , i=1,3 )
        WRITE(NUNIT2,*) CLS
        WRITE(NUNIT2,*) HOLD
        WRITE(NUNIT2,NLINE2C) ( ST(i) , i=1,3 )

c ***   Exiting the hold-screen mode
        CALL EXIT_HOLD_SCREEN_MODE( HOLD, 1, IDUMMY )
        WRITE(NUNIT2,*) HOLD
        GO TO 40
```

```
c ***   Storing Characteristics of Impulse Response in THETAOCHA.IMP
35      CONTINUE
        WRITE(NUNIT2,*) CLS
        WRITE(NUNIT2,NLINE2A) ( Thpeak_imp(i), i=1,3 ),
     +                        ( PT_imp(i), i=1,3 ),
     +                        ( ST_imp(i), i=1,3 )

c ***   Calling the Plotting routine to perform plots of the results
c       and store them into the appropiate output files.
40      CONTINUE
        CALL SCR_PLOTTER

c ***   Generation of output file for the printer: PRINTER.DAT
        REWIND 17
        page_counter = 0

50      CONTINUE
           page_counter = page_counter + 1
           WRITE(18,NLINE3) page_counter

           DO 60 i=1,50                                  ! 50 lines/record
              record_counter = record_counter + 1
              IF ( FLAG(2) )
                 THEN
                  READ(17,*,END=70) FR, TEMP1, TEMP2,
                                    ( Y(j), j=1,10 )
                  WRITE(18,950) FR, TEMP1, Y(3), Y(7),
                                    TEMP2, Y(4), Y(8),
                                     Y(1), Y(5), Y(9),
                                     Y(2), Y(6), Y(10)
                  GO TO 60
              END IF
              IF ( FLAG(4) )
                 THEN
                  READ(17,*,END=70) t             ,      ! Reading All
     +                         ( Y(j), j=1,10 )       ! States
                  WRITE(18,930) t                 ,
     +                         ( Y(j), j=1,3 ),        ! Writing All
     +                         ( Y(k), k=8,10 )        ! States

                 ELSE
                  READ(17,*,END=70) t, ( Y(j), j=1,6 )
                  WRITE(18,930) t, ( Y(j), j=1,6 )

              END IF
60         CONTINUE
        GO TO 50                          ! Return and write new page

70      CONTINUE
        RETURN

C ***************************   FORMATS   ******************************C
900     FORMAT(1H /1H /1H /1H /1H /1H /1H /1H /1H ,5X,
     +          'In order for you to see the output generated by the'/
     +   1H ,5X,'SIMULATION Program, please, PRESS the NO-SCROLL key'/
     +   1H ,5x,'to start looking at the Output...'/
     +   1H /1H /1H /1H /1H /1H /1H /1H /1H /1H )
910     FORMAT(1H /1H ,2X,'Please, Press the NO-SCROLL key to conti',
     +          'nue on next page...')
920     FORMAT( G8.3,6(G12.4) )
```

```
930     FORMAT(1H ,G12.6,6(G20.12))
940     FORMAT( G8.3, 3( 4G18.6/ 8X) )
950     FORMAT(1H , 13G10.2)

C================== FORMATS FOR STEP ANALYSIS ==========================C.
1000    FORMAT(1H ,10X,'NUMERICAL OUTPUT FOR STEP RESPONSE ANALYSIS',9x,
     +          'Page ',I3/1H ,
     +             10X,'=======================================''/1H /
     +          'Time',6X,'OMEGA(x)',4X,'OMEGA(y)',4X,'OMEGA(z)',
     +          4X,'THETA(x)',4X,'THETA(y)',4X,'THETA(z)'/'(sec)',
     +          2X,3( 3x,'(rad/sec)' ),3( 3x,'(radians)' )
     +          /4('-'),2X,6(4X,8('-')))
1010    FORMAT(1H ,20X,'Characteristics of Step Response',10X,'Page 1'
     +          /1H ,20x,32('=')/1H /1H /
     +          1H ,10X,'1.- PERCENTAGE OF MAXIMUM OVERSHOOT'
     +          ' (PMO)'/1H ,14X,37('-')/1H /
     +          1H ,14X,'PMO (x-axis) = ',G12.6,' % '/
     +          1H ,14X,'     (y-axis) = ',G12.6,' % '/
     +          1H ,14X,'     (z-axis) = ',G12.6,' % '/1H /
     +          1H ,10X,'2.- PEAK TIME (PT)'/1H ,14X,14('-')/1H /
     +          1H ,14X,' PT (x-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (y-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (z-axis) = ',G12.6,' sec. '/1H /1H /
     +          1H , 2X,'Please, Press the NO-SCROLL key to continue '
     +          'on next page...'/1H /1H )
1020    FORMAT(1H ,20X,'Characteristics of Step Response',10X,'Page 2'
     +          /1H ,20x,32('=')/1H /1H /
     +          1H ,10X,'3.- RISING TIME (RT)'/1H ,14X,16('-')/1H /
     +          1H ,14X,' RT (x-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (y-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (z-axis) = ',G12.6,' sec. '/1H /
     +          1H ,10X,'4.- DELAY TIME (DT)'/1H ,14X,15('-')/1H /
     +          1H ,14X,' DT (x-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (y-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (z-axis) = ',G12.6,' sec. '/1H /1H /
     +          1H , 2X,'Please, Press the NO-SCROLL key to continue '
     +          'on next page...'/1H /1H )
1030    FORMAT(1H ,20X,'Characteristics of Step Response',10X,'Page 3'
     +          /1H ,20x,32('=')/1H /1H /
     +          1H ,10X,'5.- SETTLING TIME (ST)'/1H ,14X,18('-')/1H /
     +          1H ,14X,' ST (x-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (y-axis) = ',G12.6,' sec. '/
     +          1H ,14X,'     (z-axis) = ',G12.6,' sec. ')
1040    FORMAT(1H1,25X,'NUMERICAL OUTPUT FOR STEP RESPONSE ANALYSIS',5x,
     +          'Page ',I3/1H ,
     +             25X,'=======================================''/1HO/
     +          1H ,2X,'Time',12X,'OMEGA(x)',12X,'OMEGA(y)',12X,'OMEGA(',
     +                 'z)',12X,'THETA(x)',12X,'THETA(y)',12X,'THETA(z)'
     +         /1H ,2X,'(sec)',3( 11X,'(rad/sec)' ),3( 11X,'(radians)' )
     +         /1H ,2X,5('-'),6(11X,9('-'))
     +         /1H )

C================== FORMATS FOR FREQUENCY ANALYSIS ====================C
2000    FORMAT(1H ,10X,'NUMERICAL OUTPUT FOR FREQUENCY RESPONSE ANALYSIS'
     +          ,5x,'Page ',I3/1H ,
     +             10X,'=============================================='
     +          /1H /1H /
     +               'Freq.    ',6X,'THETA_AMP(x)',6X,'THETA_PHA(x)',
     +                            6X,'OMEGA_AMP(x)',6X,'OMEGA_PHA(x)'/
     +                '         ',6X,'THETA_AMP(y)',6X,'THETA_PHA(y)',
```

```
     +                    6X,'OMEGA_AMP(y)',6X,'OMEGA_PHA(y)'/
     +                   ',6X,'THETA_AMP(z)',6X,'THETA_PHA(z)',
     +                    6X,'OMEGA_AMP(z)',6X,'OMEGA_PHA(z)'
     +          /1H )
2010      FORMAT(1H1,33X,'NUMERICAL OUTPUT FOR FREQUENCY RESPONSE ANALYSIS'
     +          ,5x,'Page ',I3/1H ,
     +             33X,'=================================================='
     +          /1HO/1H ,13('+---------'),'+'/1H ,
     +          '!Frequency!',21X,'ANGULAR POSITION',22X,'!',23X,
     +          'ANGULAR RATE',24X,'!'/1H ,'!',9X,'!'
     +          ,9X,'Amplitude',11X,'!',11X,'Phase',13X,'!',9X,
     +          'Amplitude',11X,'!',11X,'Phase',13X,'!'
     +          /1H ,'!',9X,'!',9X,'(Decibels)',10X,'!',9X,'(Degrees)',
     +          11X,'!',9X,'(Decibels)',10X,'!',9X,'(Degrees)',11X,'!'
     +          /1H ,'!',7X,4(2X,'! X-axis',4X,'Y-axis',4X,'Z-axis'),
     +          '  !'/1H ,13('+---------'),'+'
     +          )

C=================== FORMATS FOR IMPULSE ANALYSIS ======================C
3000      FORMAT(1H ,10X,'NUMERICAL OUTPUT FOR IMPULSE RESPONSE ANALYSIS'
     +          ,5X,'Page ',I3/1H ,
     +             10X,'============================================='
     +          /1H/1H ,'Time',6X,'OMEGA(x)',4X,'OMEGA(y)',4X,'OMEGA(z)',
     +          4X,'THETA(x)',4X,'THETA(y)',4X,'THETA(z)'/1H ,4('-'),
     +          2X,6(4X,8('-'))
     +        )
3010      FORMAT(1H ,20X,'Characteristics of Impulse Response'/1H ,20x,
     +          35('=')/1HO,10X,'ANGULAR POSITION'/1H ,10X,16('-')/
     +          1HO,10X,'1. -     Peak value (x-axis) = ',G12.6,' (rad)'/
     +          1H ,10X,'                    (y-axis) = ',G12.6,' (rad)'/
     +          1H ,10X,'                    (z-axis) = ',G12.6,' (rad)'/
     +          1HO,10X,'2. -      Peak time (x-axis) = ',G12.6,' (sec)'/
     +          1H ,10X,'                    (y-axis) = ',G12.6,' (sec)'/
     +          1H ,10X,'                    (z-axis) = ',G12.6,' (sec)'/
     +          1HO,10x,'3. - Settling time (x-axis) = ',G12.6,' (sec)'/
     +          1H ,10X,'                    (y-axis) = ',G12.6,' (sec)'/
     +          1H ,10X,'                    (z-axis) = ',G12.6,' (sec)'
     +        )
3020      FORMAT(1H1,25X,'NUMERICAL OUTPUT FOR IMPULSE RESPONSE ANALYSIS'
     +          ,5x,'Page ',I3/1H ,
     +             25X,'======================================='
     +          /1HO/
     +          1H ,2X,'Time',12X,'OMEGA(x)',12X,'OMEGA(y)',12X,'OMEGA(',
     +                 'z)',12X,'THETA(x)',12X,'THETA(y)',12X,'THETA(z)'
     +         /1H ,2X,'(sec)',3( 11X,'(rad/sec)' ),3( 11X,'(radians)' )
     +         /1H ,2X,5('-'),6(11X,9('-'))
     +         /1H )

          END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     Subroutine RUNGE_KUTTA                         *
c *                                                                    *
c **********************************************************************

c ***   Solution of a system of first order ordinary differential
c       equations with given initial values.
c       Particular situation: 6 variables devided into two sets, one for
c       position and the other for angular velocity.

c ***   If Quaternion's Computations are going to be performed a new set
c       of 4 variables is added.

c ***   The method applied is by means of 4th order Runge-Kutta formul-
c       ation by using the modification due to Gill.

c ***   Written by Juan J. Rodriguez-Moscoso on 22-Jan-85
c       Last revised on 01-Jun-85

        SUBROUTINE RUNGE_KUTTA ( ndim   ,  ! Dimension of States.
     .                           t0     ,  ! Initial time.
     .                           tf     ,  ! Final time.
     .                           tincr  ,  ! Time increment.
     .                           EPS    ,  ! Precision Epsilon.
     .                           Y      ,  ! Array of States.
     .                           Ydot   ,  ! Derivative of States.
     .                           *       ) ! Error Return code.

c--------------------------------------------------------------------
c               COMMON AREAS DEFINITIONS AND ARRAYS                  *
c--------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        DIMENSION       Y(10)           , Ydot(10)      , aux(16,10)    ,
     +                  a(4)            , b(4)          , c(4)

c--------------------------------------------------------------------
c*                      COMPUTATIONS                                 *
c--------------------------------------------------------------------

        DO i=1,ndim
           aux(8,i) = 0.0666666666666666667d0*Ydot(i)
        END DO

             t = t0             ! Starting time.
          tend = tf             ! Ending time.
        deltat = tincr          ! Time increment.
         ICODE = 0              ! Stop condition.

        CALL SYS_DYNAM ( ndim, t, Y, Ydot )
        IF ( deltat*( tend-t ) ) 380, 370, 20   ! Error Test.

20      a( 1) = 0.5d0
```

```
      a( 2) = 0.2928932188134524d0
      a( 3) = 1.7071067811865475d0
      a( 4) = 0.1666666666666667d0

      b( 1) = 2.0d0
      b( 2) = 1.0d0
      b( 3) = 1.0d0
      b( 4) = 2.0d0

      c( 1) = 0.5d0
      c( 2) = a( 2)
      c( 3) = a( 3)
      c( 4) = 0.5d0

      DO i=1,ndim              ! First Runge-Kutta step.
         aux( 1, i) = Y(i)
         aux( 2, i) = Ydot(i)
         aux( 3, i) = 0.d0
         aux( 6, i) = 0.d0
      END DO

        irec = 0
      deltat = deltat + deltat
        ihlf = -1
       istep = 0
        iend = 0

40    IF ( (t + deltat - tend)*deltat ) 70, 60, 50
50    deltat = tend - t
60      iend = 1
70    CONTINUE
      CALL OUTPUT_17 ( ndim, t, tincr, Y, Ydot )
      CALL CHECK_ERROR ( irec, ICODE )
      IF ( ICODE .NE. 0 ) RETURN 1
      itest = 0
90    istep = istep + 1

       j = 1           ! Start of innermost Runge-Kutta loop.
100   aj = a( j)
      bj = b( j)
      cj = c( j)
      DO i=1,ndim
           r1 = deltat*Ydot(i)
           r2 = aj*( r1-bj*aux(6,i) )
         Y(i) = Y(i) + r2
           r2 = r2 + r2 + r2
         aux(6,i) = aux(6,i) + r2 - cj*r1
      END DO

      IF ( j - 4 ) 120, 150, 150
120   j = j + 1
      IF ( j - 3 ) 130, 140, 130
130   t = t + 0.5d0*deltat
140   CALL SYS_DYNAM ( ndim, t, Y, Ydot )
      GO TO 100

150   IF ( itest ) 160, 160, 200      ! Test of Accurracy
160   DO i=1,ndim
         aux( 4, i) = Y(i)
      END DO
```

```
          itest = 1
          istep = istep + istep - 2

180        ihlf = ihlf + 1
              t = t - deltat
          deltat = 0.5d0*deltat
          DO i=1,ndim
                  Y(i) = aux( 1, i)
               Ydot(i) = aux( 2, i)
             aux( 6, i) = aux( 3, i)
          END DO
          GO TO 90

200       imod = istep/2
          IF ( istep - imod - imod ) 210, 230, 210
210       CONTINUE
          CALL SYS_DYNAM ( ndim, t, Y, Ydot )
          DO i=1,ndim
             aux(5,i) = Y(i)
             aux(7,i) = Ydot(i)
          END DO
          GO TO 90

230       delt = 0.0D0
          DO i=1,ndim
             delt = delt + aux(8,i)*DABS( aux(4,i)-Y(i) )
          END DO
          IF ( delt - EPS ) 280, 280, 250
250       IF ( ihlf - 10  ) 260, 360, 360
260       DO i=1,ndim
             aux( 4, i) = aux( 5, i)
          END DO
          istep = istep + istep - 4
              t = t - deltat
           iend = 0
          GO TO 180

c ***     Result values are good
280       CONTINUE
          CALL SYS_DYNAM ( ndim, t, Y, Ydot )
          DO i=1,ndim
             aux( 1, i) = Y(i)
             aux( 2, i) = Ydot(i)
             aux( 3, i) = aux( 6, i)
                Y(i) = aux( 5, i)
             Ydot(i) = aux( 7, i)
          END DO
          CALL OUTPUT_17 ( ndim, t-deltat, tincr, Y, Ydot )
          CALL CHECK_ERROR ( ihlf, ICODE )
          IF ( ICODE .NE. 0 ) RETURN 1
          DO i=1,ndim
                Y(i) = aux( 1, i)
             Ydot(i) = aux( 2, i)
          END DO
          irec = ihlf
          IF ( iend ) 320, 320, 390

320         ihlf = ihlf - 1       ! Increment gets doubled.
           istep = istep/2
          deltat = deltat + deltat
```

```
      IF ( ihlf ) 40, 330, 330

330   imod = istep/2
      IF ( istep - imod - imod ) 40, 340, 40

340   IF ( delt - 0.02d0*EPS ) 350, 350, 40
350     ihlf = ihlf - 1
       istep = istep/2
      deltat = deltat + deltat
      GO TO 40

360   ihlf = 11
      CALL SYS_DYNAM ( ndim, t, Y, Ydot )
      GO TO 390

370   ihlf = 12
      GO TO 390

380   ihlf = 13
390   CALL OUTPUT_17 ( ndim, t, tincr, Y, Ydot )
      CALL CHECK_ERROR ( ihlf, ICODE )
      IF ( ICODE .NE. 0 ) RETURN 1

      RETURN
      END
```

```
c ***************************************************************************
c *                                                                         *
c *                      Subroutine SCR_PLOTTER                             *
c *                                                                         *
c ***************************************************************************

c.....  Written by Juan J. Rodriguez-Moscoso on 07-May-85

c.....  This routine reads in a file FOR017.DAT previously created  with
c.....  all data to be plotted. Then, it proceeds to build a screen plot
c.....  of this data for VT100's like terminals.

        SUBROUTINE SCR_PLOTTER

c---------------------------------------------------------------------------
c*              COMMON AREAS AND ARRAYS DEFINITIONS                        *
c---------------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

c ***   Common Areas Definitions:
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)

        COMMON / FLAG / FLAG
        COMMON / SCRP / number_records, ! Step Response
     .                  number_decades, ! Frequency Response
     .                  number_samp_f

c ***   Dimensioning of Arrays
        DIMENSION       Y(12,61),       ! Plot of axes
     .                  Ymax(12),
     .                  Ymin(12),
     .                  t(7)

c ***   Variable definitions
        LOGICAL*1       FLAG(7)
        REAL*8          Kp      , Kd    , INMAT , INMATV
        CHARACTER*1     plot_symbol(6), blank, bode_symbol(4)
        CHARACTER*3     home_pos
        CHARACTER*4     up1, TITULO2A
        CHARACTER*5     mover, moved, curs_pos, mover10, TITULO7B,
     .                  TITULO8
        CHARACTER*7     TITULO3A, TITULO3B
        CHARACTER*8     TITULO2B
        CHARACTER*9     TITULO7A, UNIDAD(3)
        CHARACTER*10    blanco
        CHARACTER*12    TITULO4A, TITULO4B, TITULO6, TITULO9
        CHARACTER*15    TITULO1
        CHARACTER*17    TITULO5
        CHARACTER*66    TITULO

        DATA            plot_symbol/'x','y','z','x','y','z'/, blank/' '/
        DATA            TITULO1/'Plot of Angular'/,
     .                  TITULO2A/'RATE'/, TITULO2B/'POSITION'/,
     .                  TITULO3A/'[Omega('/, TITULO3B/'[Theta('/,
     .                  TITULO4A/')] (rad/sec)'/, TITULO4B/')] (radians)'/
     .                , TITULO5/'vs Time (seconds)'/,
     .                  TITULO6/'Bode Plot of'/,
     .                  TITULO7A/'AMPLITUDE'/, TITULO7B/'PHASE'/,
```

```
                    TITULO8/'-axis'/,
                    TITULO9/'vs FREQUENCY'/
      DATA          UNIDAD/'(decibel)','(degrees)','(rad/sec)'/
      DATA          TITULO/' '/, blanco/' '/
      DATA          bode_symbol/'A','P','A','P'/

c-------------------------------------------------------------------------
c*                        PLOTTING                                       *
c-------------------------------------------------------------------------
c ***   Check out FLAGs to determine how many plotting files are needed.
      REWIND 17
      call cursor_right( mover10, 1, idumm, 10 )
      call cursor_up( up1, 1, idumm, 1 )
      call cursor_home( home_pos, 1, idummy )

      DO i=1,3
         IF ( FLAG(i) )
            THEN
             GO TO ( 1, 2, 3 ), i
         END IF
      END DO

c==================== THETAOPLOT.STP & OMEGAOPLOT.STP ================c
1     CONTINUE
      is_salto = (number_records - 1)/60
      is_resto = MOD( number_records - 1, 60 )
      is_resal = is_salto
      kt = 1
      IF ( FLAG(4) )
         THEN
          READ(17,*) t(1), ( Y(k,1), k=1,3 ), TEMP, TEMP, TEMP, TEMP,
                     ( Y(k,1), k=4,6 )
          DO I = 1,60
             DO J = 1, is_resal
                READ(17,*) t(kt+1), ( Y(k,I+1), k=1,3 ), TEMP, TEMP,
                           TEMP, TEMP, ( Y(k,I+1), k=4,6 )
             END DO
             IF ( MOD(I,10) .EQ. 0 ) kt = kt + 1
          END DO
         ELSE
          READ(17,*) t(1), ( Y(k,1), k=1,6 )
          DO I = 1,60
             DO J = 1, is_resal
                READ(17,*) t(kt+1), ( Y(k,I+1), k=1,6 )
             END DO
             IF ( MOD(I,10) .EQ. 0 ) kt = kt + 1
          END DO
      END IF

      DO I = 1,6
         Ymax(I) = 0.0
         Ymin(I) = 0.0
      END DO
      DO I = 1,60
         DO J = 1,6
            IF ( Y(J,I) .GT. Ymax(J) ) Ymax(J) = Y(J,I)
            IF ( Y(J,I) .LT. Ymin(J) ) Ymin(J) = Y(J,I)
         END DO
      END DO
```

```
          iunit = 9
          DO I = 1,6          ! Plotting first OMEGAOPLT.STP for every axis,
                              ! and then THETAOPLT.STP
             TITULO = ' '
              TITULO(1:28) = TITULO1//blank//TITULO2A//blank//TITULO3A
             TITULO(29:59) = plot_symbol(i)//TITULO4A//blank//TITULO5
             IF ( I .GT. 3 )
                THEN
                 iunit = 8
                 TITULO(17:63) = TITULO2B//blank//TITULO3B//plot_symbol(i)
                               //TITULO4B//blank//TITULO5
             END IF
             call clear_display( iunit,'*' )
             call draw_axes( iunit, Ymax(I), Ymin(I), t, kt )
             WRITE(iunit,*) home_pos, blanco, TITULO
             call cursor_down( moved, 1, idummy, 21 )
             idummy = idummy - 1
             WRITE(iunit,*) moved(1:idummy), up1
             posit =  20.0*( Y(I,1) - Ymin(I) )/( Ymax(i)-Ymin(I) )+1.0D0
             new_position = NINT(posit)
             call cursor_up( moved, 1, idumm, new_position )
             idumm = idumm-1
             WRITE(iunit,*) moved(1:idumm), mover10, plot_symbol(i), up1
             old_pos = posit
             DO J=1,60
                   posit = 20.0*( Y(i,j+1)-Ymin(i) )/(Ymax(i)-Ymin(i))+1.0
                is_posic = NINT(posit) - NINT(old_pos)
                 old_pos = posit
                new_position = is_posic
                IF ( new_position .GE. 0 )
                   THEN
                    call cursor_up( curs_pos, 1, idum, new_position )
                   ELSE
                    call cursor_down( curs_pos, 1, idum,
                                      IABS(new_position) )
                END IF
                idum = idum-1
                call cursor_right( mover, 1, idumm, J )
                idumm = idumm - 1
                IF ( new_position .EQ. 0 )
                   THEN
                    WRITE(iunit,*) mover10, mover(1:idumm),
                                   plot_symbol(i), up1
                   ELSE
                    WRITE(iunit,*) curs_pos(1:idum), mover10,
                                   mover(1:idumm), plot_symbol(i), up1
                END IF
             END DO

          WRITE(iunit,*) home_pos
          call cursor_down( moved, 1, idummy, 21 )
          idummy=idummy - 1
          WRITE(iunit,*) moved(1:idummy)

          END DO
          RETURN

c=================== THETAOAMP.FRQ & THETAOPHA.FRQ ==================c
c                    OMEGAOAMP.FRQ & OMEGAOPHA.FRQ                   c
2         CONTINUE
```

```
      kt = 0
      number_records = number_decades*number_samp_f + 1
      DO i=1,number_records
         IF ( Bound(13) .EQ. 1.0D0 )
            THEN
             READ(17,*) t(kt+1), ( Y(k,i), k=1,4 ), (TEMP, k=1,8)
             iaxis = 1
            ELSE
             IF ( Bound(14) .EQ. 1.0D0 )
                THEN
                 READ(17,*) t(kt+1), (TEMP, k=1,4), ( Y(k,i), k=1,4 )
                           , ( TEMP, k=1,4 )
                 iaxis = 2
                ELSE
                 READ(17,*) t(kt+1), (TEMP, k=1,8), ( Y(k,i), k=1,4 )
                 iaxis = 3
             END IF
         END IF
         IF ( MOD( i-1,number_samp_f ) .EQ. 0 ) kt = kt + 1
      END DO
      DO i=1,4
         Ymax(i) = 0.0D0
         Ymin(i) = 0.0D0
      END DO
      DO i=1,number_records
         DO j=1,4
            IF ( Y(j,i) .GT. Ymax(j) ) Ymax(j) = Y(j,i)
            IF ( Y(j,i) .LT. Ymin(j) ) Ymin(j) = Y(j,i)
         END DO
      END DO

      DO I=1,4
         TITULO = ' '
         TITULO(1:13) = TITULO6//blank
         TITULO(14:22) = TITULO2B//blank
         IF ( I .GT. 2 ) TITULO(14:22) = '  '//TITULO2A//''    '
         IF ( MOD(I-1,2) .EQ. 0 )
            THEN
              TITULO(23:36) = TITULO7A//'' for ''
              TITULO(37:55) = plot_symbol(iaxis)//TITULO8//blank//
                              TITULO9
              blanco = UNIDAD(1)
            ELSE
              TITULO(23:32) = TITULO7B//'' for ''
              TITULO(33:51) = plot_symbol(iaxis)//TITULO8//blank//
                              TITULO9
              blanco = UNIDAD(2)
         END IF
         iunit = 9 + I
         call clear_display( iunit,'*' )
         call draw_axes( iunit, Ymax(I), Ymin(I), t, kt )
         WRITE(iunit,*) home_pos, blanco, '    ', TITULO(1:55)
         blanco = ' '
         call cursor_down( moved, 1, idummy, 20 )
         call cursor_right( mover, 1, idum, 71 )
         idummy = idummy - 1
           idum = idum - 1
         WRITE(iunit,*) moved(1:idummy), mover(1:idumm), UNIDAD(3),
                        home_pos
         call cursor_down( moved, 1, idummy, 20 )
```

```
            idummy = idummy - 1
            WRITE(iunit,*) moved(1:idummy)
            posit =  20.0*( Y(I,1) - Ymin(I) )/( Ymax(I)-Ymin(I) )+1.0d0
            new_position = NINT(posit)
            call cursor_up( moved, 1, idumm, new_position )
            idumm = idumm-1
            WRITE(iunit,*) moved(1:idumm), mover10, bode_symbol(I), up1
            old_pos = posit

            DO J=1,number_records-1
                  posit = 20.0*( Y(i,j+1)-Ymin(i) )/(Ymax(i)-Ymin(i))+1.0
               is_posic = NINT(posit) - NINT(old_pos)
                old_pos = posit
               new_position = is_posic
               IF ( new_position .GE. 0 )
                  THEN
                   call cursor_up( curs_pos, 1, idum, new_position )
                  ELSE
                   call cursor_down( curs_pos, 1, idum,
                                     IABS(new_position) )
               END IF
               idum = idum - 1
                  K = 60*J/( number_records - 1)
               call cursor_right( mover, 1, idumm, K )
               idumm = idumm - 1
               IF ( new_position .EQ. 0 )
                  THEN
                   WRITE(iunit,*) mover10, mover(1:idumm),
                                  bode_symbol(i), up1
                  ELSE
                   WRITE(iunit,*) curs_pos(1:idum), mover10,
                                  mover(1:idumm), bode_symbol(i), up1
               END IF
            END DO
         WRITE(iunit,*) home_pos
         call cursor_down( moved, 1, idummy, 21 )
         idummy = idummy - 1
         WRITE(iunit,*) moved(1:idummy)
         END DO

         RETURN

c=================== THETAOPLT.IMP & OMEGAOPLT.IMP ====================c
3        CONTINUE
c....    Non implementation for IMPULSE RESPONSE Plotting.
         RETURN

         END
```

```
c +--------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center         !
c !                        Hunstville, Alabama                         !
c !                              1985                                  !
c +--------------------------------------------------------------------+

c **********************************************************************
c *                                                                    *
c *                     Subroutine SIMULATION                          *
c *                                                                    *
c **********************************************************************

c ***   Written by Bor-Jau Hsieh (Andy) &
c                  Juan J. Rodriguez-Moscoso on 03-Jun-85

      SUBROUTINE SIMULATION ( ICODE )

c--------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                   *
c--------------------------------------------------------------------
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      COMMON / FLAG / FLAG(7)
      COMMON / USER / user_choice
      COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                Thcom(3)        , Omcom(3)
      COMMON / STEP / PMO(3)          , RTini(3)      , RTend(3)      ,
     +                DT(3)           , ST(3)         , PT(3)         ,
     +                RT(3)           , RTimin(3)     , RTemin(3)     ,
     +                DTmin(3)        , Thpeak(3)
      COMMON / FRQ1 / Amp             , Omega         , Phase         ,
     +                ts
      COMMON / FRQ2 / THAMP(3)        , THFEE(3)      , THAOL(3)      ,
     +                THPOL(3)        , THS(3)        , THC(3)        ,
     +                X1(3)           , X2(3)         ,
     +                OMAMP(3)        , OMFEE(3)      , OMAOL(3)      ,
     +                OMPOL(3)        , OMS(3)        , OMC(3)        ,
     +                X3(3)           , X4(3)         , XN
      COMMON / FRQ3 / Freq            , Const         , Tau
      COMMON / IMPU / ST_imp(3)       , PT_imp(3)     , Thpeak_imp(3)
      COMMON / DAMP / damp_flag

      DIMENSION       Y(10)           , Ydot(10)

      INTEGER         user_choice, damp_flag
      REAL*8          Kp      , Kd    , INMAT , INMATV
      LOGICAL*1       FLAG
      CHARACTER*9     Response_type
      DATA            Response_type/' '/

c--------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c--------------------------------------------------------------------
c ***   Initializing of the common input parameters.
                      ndim = 6              ! dimension of the system
                        t0 = Bound(1)       ! initial time
                        tf = Bound(2)       ! final time
                    deltat = Bound(3)       ! time increment
```

```
c ***   Check if damping exits.
        damp_flag = 0
        DO I=1,3
           DO J=1,3
              IF ( Kd(I,J) .NE. 0.0D0 )
     +           THEN
                 damp_flag = 1
                 GOTO 10
              END IF
           END DO
        END DO
10      CONTINUE

c ***   Computing the inverse of the Inertia Matrix.
        DO I=1,3
           DO J=1,3
              INMATV(I,J) = INMAT(I,J)
           END DO
        END DO
        CALL MATINV( INMATV, 3, ICODE )
        IF ( ICODE .NE. 0 ) RETURN

c ***   Checking FLAG(i) [i=1,3] for determining type of analysis
        IF ( FLAG(1) ) THEN
c ***   STEP Response Analysis
c       ----------------------

1          CONTINUE
c -->      Setting Input Commands to Step.              ! May-28-85
           DO j=1,3
              Thcom(j) = Bound(j+12)
              Omcom(j) = Bound(j+9)
           END DO

c -->      Initialization for step response characteristics analysis
           DO j=1, 3
              RTimin(j) = Thcom(j)
              RTemin(j) = Thcom(j)
               DTmin(j) = Thcom(j)
                 PMO(j) = 0.0D0
                  RT(j) = 0.0D0
                  DT(j) = 0.0D0
                  ST(j) = 0.0D0
           END DO

           Nrt = 1
           Response_type = 'STEP'
           GOTO 100
        END IF

c ***   Frequency Response Analysis
c       ---------------------------
        IF ( FLAG(2) ) THEN
c -->      Initialize input parameters
           DO j=1,3
              Thcom(j) = 0.0d0     ! Input Commands = 0.0 for
              Omcom(j) = 0.0d0     ! Pseudo Open-Loop Frequency Response
           END DO
                 Tau = Bound(2)          ! Time constant
                 Amp = Bound(8)          ! Amplitude
```

```
               Phase = Bound(10)          ! Phase
                Ndec = Bound(11)          ! # of decades
                 Nsd = Bound(12)          ! # of sampling freq./decade
              deltat = Bound(3)
               Const = DEXP( (DLOG(10.0D0) )/DFLOAT(Nsd) )
                Freq = Bound(9)/Const     ! Lowest angular Freq
                 Nsf = Ndec*Nsd + 1
                 Nrt = Nsf
              Response_type = 'FREQUENCY'
           GOTO 100
        END IF

c ***   IMPULSE Response Analysis
c       -------------------------
        IF ( FLAG(3) ) THEN
           Nrt = 2
           Response_type = 'IMPULSE'
        END IF

c ============================== STARTING SIMULATION ==========================
100     CONTINUE
        WRITE(6,1000) Response_type
c       WRITE(6,1010)
c       READ(5,*) user_choice
        user_choice = 1          !This will be taken out.
        DO i=1, Nrt
           CALL INITIALIZE  ( Nrt       ,
     .                        ndim      ,
     .                        t0        ,
     .                        tf        ,
     .                        deltat    ,
     .                        Y          )
           CALL INTEGRATION ( ndim      ,
     .                        t0        ,
     .                        tf        ,
     .                        deltat    ,
     .                        Y         ,
     .                        Ydot      ,
     .                        &110      ,
     .                        &120       )
           CALL POLFRCAL ( deltat )
        END DO
        CALL RESULTS( Y )
        RETURN

110     ICODE = 2
        RETURN
120     ICODE = 3
        RETURN

1000    FORMAT(1H0,5X,
     .       55HFORTRAN-77 'Generic Simulation Program' under Execution
     .         /1H ,5X,55('-')/1H0,5X,' - Type of response analysis se',
     .         'lected = ',A )
c1010   FORMAT(1H ,5X,' - Would you like to see partial results during ',
c     .         'the execution '/1H ,9X,'of the simulation program?'/1H ,
c     .         20X,'1) yes'/1H ,20X,'2) no'/1H ,9X,'Select your choice:'
c     .         /1H ,12X,'# '$)
        END
```

```
c ------------------------------------------------------------------------
c |            Copyright (c) NASA Marshall Space Flight Center           |
c |                        Hunstville, Alabama                           |
c |                              1985                                    |
c ------------------------------------------------------------------------

c ************************************************************************
c *                                                                      *
c *                      SUBROUTINE SUM_ERRORS                           *
c *                                                                      *
c ************************************************************************

c       This routine calculates the errors between
c          INPUT COMMANDS and OUTPUTS
c       at summing junction of the system block diagram

c ***   Written by Juan J. Rodriguez-Moscoso &
c                  Bor-Jau Hsieh (Andy) on 04-May-85
c       Last revised on 13-May-85

        SUBROUTINE SUM_ERRORS( ndim, Thcom, Omcom, Y, Therr, Omerr )

c-----------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                       *
c-----------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)

c ***   Common Areas Definitions:
        COMMON / FLAG / FLAG(7)

c ***   Dimensioning of Arrays
        DIMENSION       Y(10),          ! Vehicle Theta & Omega
     +                  Thcom(3),       ! Input Command Position
     +                  Omcom(3)        ! Input Command Rate

        DIMENSION       Therr(3),       ! Position Error
     +                  Omerr(3)        ! Rate Error

c ***   Variables Definitions
        LOGICAL*1       FLAG

c-----------------------------------------------------------------------
c*                      COMPUTATIONS                                    *
c-----------------------------------------------------------------------
c ***   If the Quaternion block is considered during the SIMULATION, then
c       the Computation of Errors are performed only for the angular Pos-
c       ition of the vehicle.
        IF ( .NOT. FLAG(4) ) GOTO 100
        DO i = 1,3
           Therr(i) = Thcom(i) - Y(i+7)   ! Ang. pos. from Y(8) to Y(10).
           Omerr(i) = - Y(i)              ! Unvariable Omega_v.
        END DO
        RETURN

c ***   Compute Angular Position & Rate Errors for Prototype I system.
100     CONTINUE
        DO j = 1,3
           Therr(j) = Thcom(j) - Y(j+3)
           Omerr(j) = Omcom(j) - Y(j)
        END DO
```

```
RETURN
END
```

```
c +-------------------------------------------------------------------+
c !            Copyright (c) NASA Marshall Space Flight Center        !
c !                        Hunstville, Alabama                        !
c !                              1985                                 !
c +-------------------------------------------------------------------+

c ********************************************************************
c *                                                                  *
c *                     Subroutine SYS_DYNAM                         *
c *                                                                  *
c ********************************************************************

c ***   Written by Juan J. Rodriguez-Moscoso &
c                  Bor-Jau Hsieh (Andy) on 30-Apr-85
c       Last revised on 29-May-85

        SUBROUTINE SYS_DYNAM ( ndim, t, Y, Ydot )

c-------------------------------------------------------------------
c*              COMMON AREAS AND DIMENSION ARRAYS                   *
c-------------------------------------------------------------------
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON / FLAG / FLAG(7)
        COMMON /READIN/ Bound(15)       , Yin(10)       , Kp(3,3)       ,
     +                  Kd(3,3)         , INMAT(3,3)    , INMATV(3,3)   ,
     +                  Thcom(3)        , Omcom(3)

        DIMENSION       Y(10)           , Ydot(10)      ,Therr(3)       ,
     +                  Omerr(3)        , Torque(3)

        REAL*8          Kp      , Kd    , INMAT , INMATV
        LOGICAL*1       FLAG

c-------------------------------------------------------------------
c*                      COMPUTATIONS                                *
c-------------------------------------------------------------------

        CALL SUM_ERRORS ( ndim, Thcom, Omcom, Y, Therr, Omerr )
        CALL ANALYSIS   ( ndim, t, Y, Therr, Omerr )
        CALL CONTROLLER ( ndim, Therr, Omerr, Torque )
        CALL BODY_DYNAM ( ndim, Torque, Y, Ydot )
        CALL QUATERNION ( ndim, Y, Ydot )
        CALL NEW_VALUES ( ndim, Y )

        RETURN
        END
```

End of Document